AD-A248 086

DTIC
ELECTE
APR 01 1992
S D
D

MULTILAYER PERCEPTRONS FOR CLASSIFICATION

THESIS

Lisa M. Belue
Captain, USAF

AFIT/GOR/ENS/92M-02

This document has been approved for public release and sale; its distribution is unlimited.

92-08134

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright Patterson Air Force Base, Ohio

92 3 31 077

AFIT/GOR/ENS/92M-02

DTIC
ELECTE
APR 0 1 1992
S D
D

MULTILAYER PERCEPTRONS FOR CLASSFIFCATION

THESIS

Lisa M. Belue
Captain, USAF

AFIT/GOR/ENS/92M-02

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

QUALITY INSPECTED
3

Approved for public release; distribution unlimited

AFIT/GOR/ENS/92M-02

AN INVESTIGATION OF MULTILAYER PERCEPTRONS FOR CLASSIFICATION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Lisa M. Belue, B.S.
Captain, USAF

March, 1992

Approved for public release; distribution unlimited

THESIS APPROVAL

STUDENT: Captain Lisa M. Belue CLASS: GOR 92-M

THESIS TITLE: An Investigation of Multilayer Perceptrons for Classification

DEFENSE DATE: 5 MAR 92

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Advisor	Major Kenneth W. Bauer/ENS	Kenneth W Bauer Jr.
Reader	Captain Dennis W. Ruck/ENG	Dennis W. Ruck

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1992	Master's Thesis

4. TITLE AND SUBTITLE
AN INVESTIGATION OF MULTILAYER PERCEPTRONS FOR CLASSIFICATION

5. FUNDING NUMBERS

6. AUTHOR(S)
Lisa M. Belue, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/92M-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF/DPXA
Rm 5C 372
The Pentagon
Washington, D.C. 20330-5060

WL/FIVS
Wright-Patterson AFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Techniques for training, testing, and validating multilayer perceptrons are thoroughly examined. Results obtained using perceptrons are compared and contrasted with two multivariate discriminant analysis techniques - logistic regression and k-nearest neighbor. Methods for determining significant input features are investigated and a procedure for examining the confidence to place in the significance of these features is developed. Procedures to evaluate the applicability of high-order feature inputs are examined. These methods and procedures are applied to two very different applications. The first application concerns the prediction of Air Force pilot retention/separation rates for input to force projection models. The second application concerns the classification of Armor Piercing Incendiary (API) projectiles based on firing parameters. Results showed that none of the classification methods considered was able to accurately classify individual pilot's retention decisions, however, multilayer perceptrons were judged to be the superior discriminator for the classification of API projectiles. For the API projectile analysis, a procedure to determine which input features are no more significant than noise was demonstrated. The resulting salient set of feature inputs was shown to train quicker and decrease the output error. A method to identify appropriate high-order inputs was also demonstrated.

14. SUBJECT TERMS
Neural networks, Pattern recognition, Discriminant analysis, Incendiary projectiles, Pilots

15. NUMBER OF PAGES
204

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Preface

The purpose of this research effort was to investigate multilayer perceptons as classifiers and predictors. To provide a baseline for the performance of the perceptrons, the perceptron method was compared to two other classifiers--logistic regression and k-nearest-neighbor. The focus this investigation was on the application of discrimination methods on two particular applications—the discrimination of Air Force pilots in terms of their decisions to remain in military service and classification of the performance of Armor Piercing Incendiary (API) projectiles. The analysis conducted on these applications included the following investigation areas:

- Procedures for finding the appropriate architecture for a multilayer perceptron with regard to the specific application.
- The performance of discriminators when the features are categorical and binary.
- Methods for determining significant features.
- A method to determine the confidence to place on the significance of the features.
- A method for determining the appropriateness of high-order features.

In performing this analysis, writing the computer code, and compiling this thesis, I have had a great deal of help from certain specific individuals. First, I am very thankful that I chose Maj K.W. Bauer as my faculty advisor. I am greatful that he had the patience to listen to my ramblings and offer meaningful suggestions. I am also indebted to Capt D.W. Ruck for his quiet but pointed suggestions that kept me moving in the right direction. Special thanks goes to Capt Jean Steppe for her words of encouragement and sound advice at the times when I needed it most. Finally, I'd like to thank my husband Ken and my daughter Caitlin for their concern for me and for their ability to keep things going while Mommy was at school.

Lisa M. Belue

Table of Contents

List of Figures

List of Tables

AFIT/GOR/ENS/92M-02

Abstract

Techniques for training, testing, and validating multilayer perceptrons are thoroughly examined. Results obtained using perceptrons are compared and contrasted with two multivariate discriminant analysis techniques—logistic regression and k-nearest neighbor. Methods for determining significant input features are investigated and a procedure for examining the confidence to place in the significance of these features is developed. Procedures to evaluate the applicability of high-order feature inputs are examined. These methods and procedures are applied to two very different applications. The first application concerns the prediction of Air Force pilot retention/separation rates for input to force projection models. The second application concerns the classification of Armor Piercing Incendiary (API) projectiles based on firing parameters. Results showed that none of the classification methods considered was able to accurately classify individual pilot's retention decisions, however, multilayer perceptrons were judged to be the superior discriminator for the classification of API projectiles. For the API projectile analysis, a procedure to determine which input features are no more significant than noise was demonstrated. The resulting salient set of feature inputs was shown to train quicker and decrease the output error. A method to identify appropriate high-order inputs was also demonstrated.

AN INVESTIGATION OF MULTILAYER PERCEPTRONS FOR CLASSIFICATION

I. Introduction

This research effort explores the use of multilayer perceptrons for the classification and prediction of outcomes for two very different applications.

In the first application, methodologies for predicting pilot retention/separation rates for input to personnel inventory projection models were explored. Specifically, the multilayer perceptron technique was investigated with the multivariate discriminant analysis techniques used as a benchmark for the results. The objective was to determine an individual's career decision based on his individual attributes. This comparative analysis was sponsored by the Analysis Division of the Directorate of Personnel Plans, Headquarters Air Force (AF/DPXA) and the results will be incorporated into their current work on overall personnel inventory analysis.

In the second application, methodologies for classifying the functioning quality of armor piercing incendiary (API) projectiles that impact one material were investigated. The application of neural networks for classifying the performance of the projectile based on the parameters of the test shots was developed. Also, the multilayer perceptron methodology was compared to multivariate discriminant analysis techniques. This portion of the analysis was sponsored by the Wright Laboratories Survivability Enhancement Branch, Wright Patterson Air Force Base and is the topic of an entire thesis titled *Predicting Armor Piercing Incendiary Projectile Effects After Impacting Composite Material*(15).

Through these extremely varied applications, several aspects of the use of neural networks in comparison to traditional multivariate discriminant analysis were

revealed. For example, the classification of Air Force pilots was based on categorical binary data while the classification of the performance of projectiles involved continuous, measured data. Each application offered insights into the intracacies of the overall problem of group discrimination. Specific aspects that were investigated include

- Procedures for finding the appropriate architecture for a multilayer perceptron with regard to the specific application.
- The performance of discriminators when the features are categorical and binary.
- Methods for determining significant features.
- A method to determine the confidence to place on the significance of the features.
- A method for determining the appropriateness of high-order features.

The following paragraphs provide the background, the research objectives, and scope of this study.

1.1 Background

1.1.1 Discriminant Analysis Techniques Only recently have neural networks come to the forefront of discriminant analysis methods. Statistical discriminant analysis methods such as Fisher's method, k-nearest-neighbor and logistic regression have often been dismissed as not useful since the determination of their parameters was difficult or they were difficult to apply once they were derived. Neural networks, and more specifically multilayer perceptrons, have the advantage of learning their optimal parameters and are simple to apply once these parameters are found. There is also an appealing quality to the "brain-like" structure of neural networks that has caused attentions to be turned to these relatively new classifiers.

The use of neural networks has not been explored deeply from a mathematical viewpoint. Methods for determining the optimal structure of a multilayer perceptron, determining significant features, and for estimating the error rate have all been suggested, but very few formal rules have been established.

1.1.2 Classifying Pilot's Retention Decisions The Directorate of Personnel Plans, Headquarters Air Force (AF/DPX) is responsible for overall planning and policy determination for Air Force personnel. In support of this mission, the Analysis Division develops and maintains force projection models to analyze the effects of compensations and policy alternatives (12:1). The results of these projection models are highly dependent on the retention rates that are input to them. If the retention rates are inaccurate, predictions on the future structure of the officer and enlisted force may be inaccurate.

The Air Force Council, under the leadership of the Air Force Chief of Staff, recently (December 1990) directed AF/DPXA to "develop a model for the dynamics of the pilot management problem" (1). Currently, AF/DPXA uses a logit method to predict these essential retention/separation rates. However, this method of prediction is not always entirely reliable and a more accurate method must be found.

Air Force pilots begin their careers in one of three commissioning programs, the Air Force Academy, Reserve Officer Training Corps or Officer Training School. They are trained in Undergraduate Pilot Training (UPT) which lasts approximately one year. These pilots are obligated to four, five or six years, depending on their date of entry to pilot training. After completing UPT, the individual attends advanced training in a specific aircraft. At the completion of advanced training, pilots are assigned to operational units where they perform primary flight crew duties. After an initial operational assignment, pilots can expect to continue flying in their original weapon system, be assigned to a staff position, serve as an instructor pilot, or obtain an advanced degree. In addition, the individual may receive assignments to long or

short tour locations overseas. During this time, the officer continues to incur service obligations, for example, every time he completes training or moves to a new base. At the completion of all service obligations, the officer is free to separate if he so desires.

Each individual pilot weighs the benefits of staying in the Air Force with the benefits of leaving and seeking civilian employment. Several elements enter into each individual's decision of whether to remain or separate. Military pay may be one of the greatest factors in the decision. The health of the civilian sector and civilian airline hirings might also have a significant influence. Another potential factor may be that individuals feel as though they have little say in their future assignments.

Each individual must also weigh the effects that staying in the military will have on their family. Spouses may be unable to find employment in certain locations. The pilot's duties may make him absent from the home too often. The factors that enter into the separation decision are as varied as the individuals that must make the decisions.

This research effort analyzed the *individual* attributes of Air Force pilots and attempts to predict the decisions they will make. The resulting method for classifying the retention/separation of pilots will greatly improve the management of this very critical, expensive resource.

1.1.3 Predicting API Projectile Effects The Air Force's Computation of Vulnerable Areas and Repair Times (COVART) software serves as a primary tool for aircraft vulnerability analysis. COVART is used by the Air Force's Aircraft Survivability Research Facility, aircraft Special Program Offices and numerous logistics centers. The COVART code incorporates impacting projectile effects into its computation of aircraft damage.

For the past several years, the Survivability Enhancement Branch at Wright Patterson Air Force Base has analyzed the penetration prediction equations for API

projectiles impacting materials. Their research shows that the current methods for determining the incendiary functioning of the projectiles are inaccurate and therefore, aircraft damage analyses will also be inaccurate.

The cause of the inaccuracies in the COVART simulation stem primarily from the use of test data that does not include shots into composite materials. Instead, it was assumed that aluminum properties would accurately predict an API projectile effects upon impact of a composite. This research effort investigates ways to accurately classify API projectile's performance based on the parameters of the firing. For each given shot, four independent variables are known; impact or striking velocity (V_S), impact or striking mass (M_S), panel ply thickness in inches ($TKIN$), and the impact obliquity angle (OBL). In this analysis, OBL is converted to the secant of the angle ($SECT$), a commonly accepted practice in penetration mechanics' analysis.

1.2 Research Objectives

The purpose of this research was to investigate the overall use of multilayer perceptrons for classification. Research objectives were established in the following areas:

- Development of Applicable Decision Factors
- Data Collection and Orientation
- Development of Classification Methodology
- Feature Selection
- Comparisons with other Classifiers
- Applications to Specific Problems

In order to investigate each of these objectives, two separate applications were used—classifying Air Force pilots retention decisions and classifying API projectile firing performance. Specifically, the following objectives were established.

1.2.1 Development of Applicable Decision Factors The first objective was an investigation of the methods for determining the candidate factors that should be considered when developing a classifier. For classifying pilots, the specific objective was to identify significant factors that enter into the decisions of pilots as to whether they should remain with or leave military service. The goal was to find a list of at least twenty potentially relevant factors that were captured in the personnel files maintained by the Air Force. For classifying API projectile firings, the data supplied by the Survivability Enhancement Branch was the only data available to the analyst and, therefore, this research objective is not applicable.

1.2.2 Data Collection and Orientation The next objective concerned the collection and orientation of the data for use in deriving both the multilayer perceptron discriminator and the traditional multivariate discriminators. For both applications, the sponsors provided the data. However, all classification procedures require input features to be in a numeric form (as opposed to categorical). For example, the attributes of Air Force pilots were primarily categorical. A goal was the proper translation of this categorical data into a binary format.

1.2.3 Development of Classification Methodologies A third objective was to actually develop classifiers for the two applications. Researchers have investigated many structures of multilayer perceptrons, and the algorithms to implement their use. Whether a structure works well or not is often dependent on the problem itself (21:61) Therefore, a goal was the investigation of multilayer perceptron structures and algorithms to find the optimal configuration for the specific applications of classifying Air Force pilots and classifying API projectile firings. Several tradi-

tional multivariate methods are available for predicting responses given the factors affecting the response (5:360)(24:42). A secondary goal was to investigate the logistic regression technique and the k-nearest-neighbor technique as alternatives to the multilayer perceptron discriminator.

1.2.4 Feature Selection. Another objective was to investigate formal methods for determining those factors that are most significant in the development of the specific classifier. The goal of this objective was to reduce the number of features used for each classification technique and still obtain accurate, complete results in each prediction methodology. Two secondary goals were to compare feature selection procedures for the multilayer perceptron technique with discriminant analysis feature selection techniques and investigate procedures for determining the confidence to place in the significance of a feature.

1.2.5 Comparison of Results. The next objective of the study was to determine the preferred method of classification. A goal was to estimate the actual error rates for each of the methods. In addition, a goal was to produce a subjective comparison of the computational complexity of each method.

1.2.6 Application of Techniques to Specific Problems. An final objective was to apply discrimination techniques judged to be most accurate to the application areas. The goal concerning the first application was to investigate the applicability of multilayer perceptrons and multivariate discriminant analysis techniques for classifying an individual pilot's decision as "stay" of "leave" during the next year based on the characteristics of the individual. The rates produced by the most accurate procedure will then be used by personnel analysts in the development of personnel inventory models. Similarly, the goal concerning the second application was to investigate classification procedures to classify a projectile functions as "complete" or "other" based on the parameters of the shot. The rates produced by the method

judged to be most accurate will be used as inputs to models such as the COVART simulation.

1.3 Scope

This thesis develops both the multilayer perceptron technique and two discriminant analysis techniques and then compares their applicability. Further, a general discussion is offered of past attempts to predict pilot retention and projectile functioning, an overview of multilayer perceptrons and an overview of the two discriminant analysis techniques used. Appendices include

1. The code for a multilayer perceptron written in FORTRAN 77 containing subroutines for the following purposes
    - Data Input
    - Data Normalization
    - Multilayer Perceptron Training
    - Output Analysis
    - Feature Selection
    - Evaluating High-Order Inputs
2. Statistical Application Software (SAS) routines for the development of the logistic regression and k-nearest-neighbor multivariate discriminant techniques,
3. FORTRAN 77 programs and SAS routines for feature selection,
4. FORTRAN 77 programs for data translation and orientation.

II. Literature Review

This chapter provides a review of the literature concerning the areas of multilayer perceptron analysis, multivariate discriminant analysis techniques, error rate computations, retention rate prediction, and API projectile firing prediction. The intent is to give the general background of each topic area and, to highlight any aspects of these subjects applicable to this investigation.

2.1 Multilayer Perceptrons

This literature review describes certain aspects of multilayer perceptrons as they apply to calssification and prediction. Specifically, these aspects include the general structure, the network architecture, two implementation algorithms, the data requirements, and the input features.

2.1.1 Multilayer Perceptron Terms Defined. Because neural networks are a relatively new technique for discrimination, several terms particular to these networks are defined below.

- **Backpropagation** A learning algorithm for updating weights in a multilayer, feedforward, mapping neural network that minimizes mean squared mapping error (3).
- **Epoch** A complete presentation of the data set being used to train the multilayer perceptron. Also called a training cycle.
- **Feature** In neural networks, the term feature is used to define a measurement which is made on input vectors which contains information useful for distinguishing the various classes.
- **Feedforward** Characterized by multilayer neural networks whose connections exclusively feed inputs from lower to higher layers; in contrast to a feedback

network, a feedforward network operates only until its inputs propagate to its output layer. An example of a feedforward neural network is the multilayer perceptron (3).

- **Hidden Units** Those processing elements in multilayer neural network architectures which are neither the input layer nor the output layer, but are located in between these and allow the network to undertake more complex problem solving (3).
- **Learning Algoithms** In neural networks, the equations which modify some of the weights of processing elements in response to input and output values (3).
- **Multilayer Perceptron** A multilayer feedforward network that is fully connected and which is typically trained by the backpropagation learning algorithm (3).
- **Neural Network** An information processing system which operates on inputs to extract information and produces outputs corresponding to the extracted information (3).
- **Single-layer Perceptron** A type of neural network algorithm used in pattern classification problems and trained with supervision. The single-layer perceptron generated much interest when it was initially developed in the 1950s by Rosenblatt because of its ability to learn to recognize simple patterns. Connection weights and thresholds in a perceptron can be fixed or adapted using a number of different algorithms (3).
- **Supervised Training** A means of training adaptive neural networks which requires labeled training data and an external teacher. The teacher knows the correct response and provides an error signal when an error is made by the network (3).

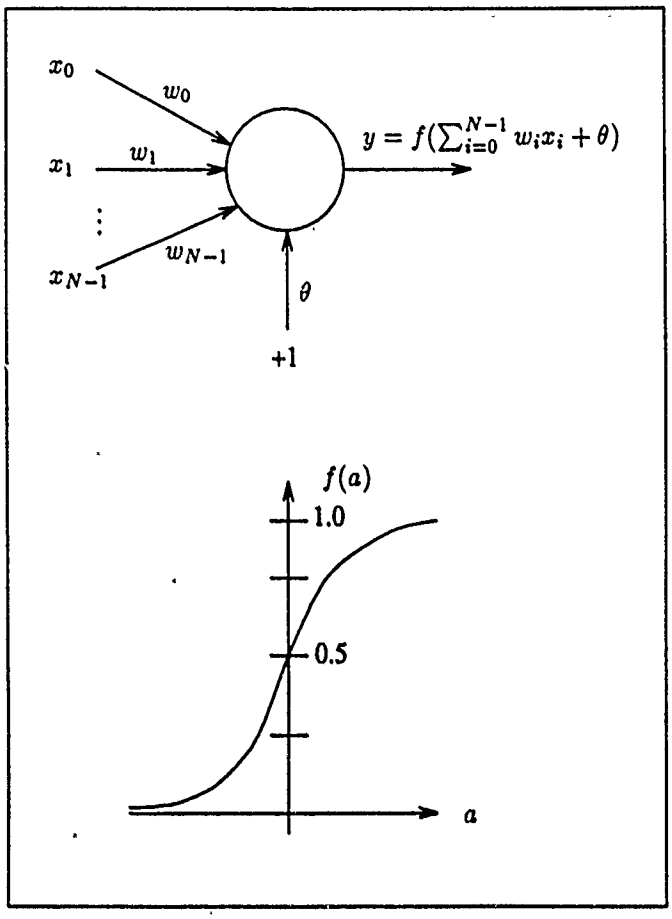


Figure 1. Single Perceptron

Reprinted from (21:48)

- **Weight** A processing element (or neuron or unit) need not treat all inputs uniformly. Processing elements receive inputs by means of interconnects (also called 'connections' or 'links'); each of these connections has an associated weight which signifies its strength. The weights are combined to calculate the activations (3).

2.1.2 Description of Multilayer Perceptrons. Figure 1 shows a single perceptron. Rogers attributes this architecture to Rosenblatt in *Principles of Neurodynamics* published in 1959 (21:47).

Data feeds into the perceptron's input nodes numbered x_0 to x_{N-1} and the w_i on each branch of the perceptron weights the inputs. The procedure sums across the

weighted inputs, adds a bias term, and transforms the sum so that the activation y of the perceptron is:

$$y = f_h[(\sum_{i=0}^{N-1} w_i x_i) + \theta] \tag{1}$$

The bias, or threshold is an additional node added to each layer of a multilayer perceptron whose input is one. Therefore, the threshold times the weight connecting the threshold to the next layer is a constant. The nonlinear transformation $f_h[\cdot]$ most often takes the form of a sigmoid; however, it could be a hard limiter or a linear ramp. (See Figure 2.)

For each input, the single perceptron outputs a single value that signifies the classification of the input (21:49). Training the perceptron to classify inputs consists of finding the weights that produce outputs near "1" for one class and near "0" for the other class.

According to Minsky and Papert, the single layer perceptron does not allow discrimination between classes that are not hyperplane separable (18:249-252). Beginning in the 1980's, researchers developed methods for layering the single perceptron to allow for complex, nonlinear boundaries between classes (22:13). Figure 3 shows a three layer perceptron. (Note that the numbering scheme counts only the "hidden" layers between input and output.) Rogers says Cybenko has shown "only one hidden layer is sufficient for any arbitrary transformation, given enough nodes" (21:53).

Multilayer perceptrons have two major advantages over more traditional discriminant analysis techniques. First, multilayer perceptrons do not require assumptions as to the distribution of the data or the equality of the covariance matrices of the groups of data to be classified. Second, multilayer perceptrons allow for the formation of nonlinear decisions regions, including disjoint regions.

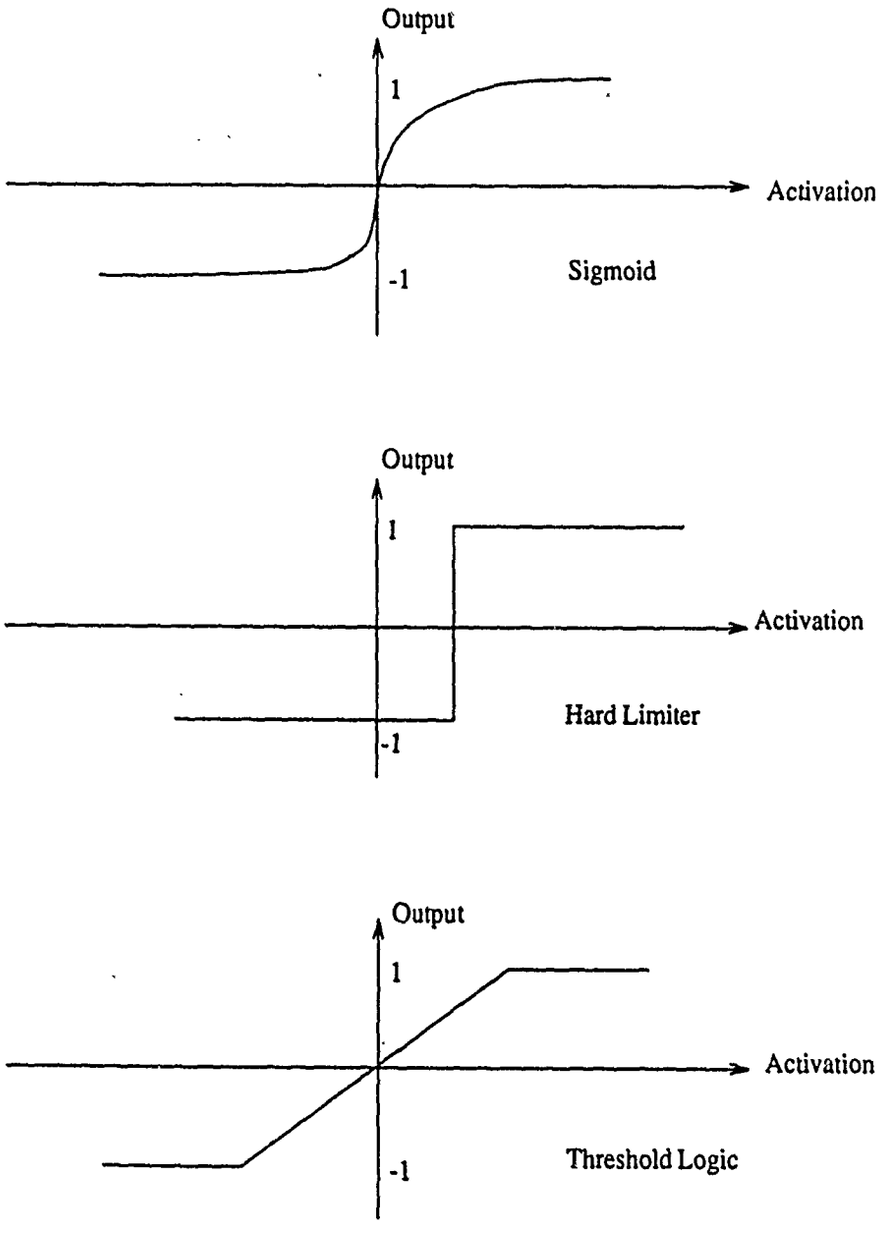


Figure 2. Sigmoid, Hard Limiter and Linear Ramp
Reprinted from (21:50)

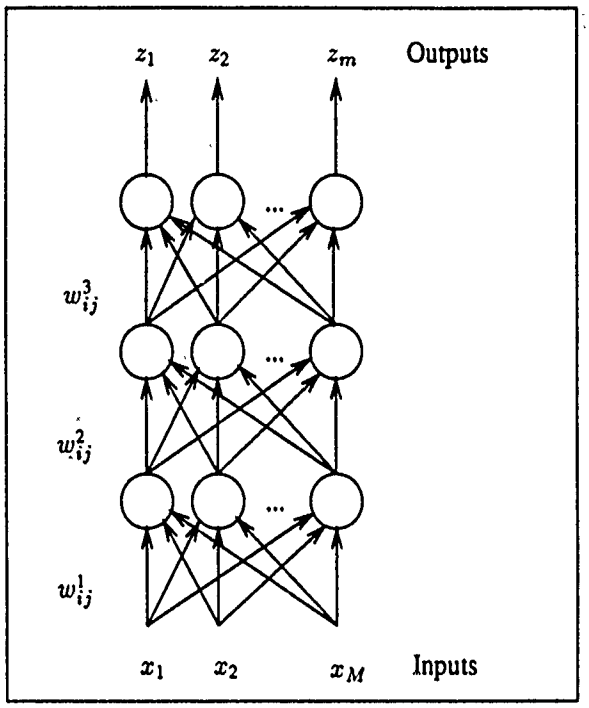


Figure 3. Multilayer Perceptron

Reprinted from (21:55)

2.1.3 Structure of the Network. The input layer of a multilayer perceptron will have as many nodes as there are features. The output layer will normally have one node for every class of outputs. Consequently, the structure of multilayer perceptrons varies only in the number of hidden layers and the number of nodes in each of these hidden layers. Ruck states: "Rigorous mathematical techniques have not been developed to determine the appropriate number of hidden layers or the number of nodes in those layers for a given problem" (22:97).

Ruck does, however, provide a heuristic for sizing a multilayer perceptron. First, his technique requires a baseline performance level. Next, the technique steps through the possible multilayer perceptron architectures, investigating the following potential components:

1. no hidden layers
2. a single layer with five nodes

3. a single layer with more than five nodes
4. a second hidden layer with five nodes
5. more nodes in the first hidden layer
6. more nodes in the second hidden layer (22:97-99).

The heuristic concludes with the selection of the most promising architectures and the training of these networks until performance stabilizes. Following the above procedure, a user should be able to construct a multilayer perceptron architecture that produces results close to the performance baseline.

2.1.4 Backpropagation Algorithm. Training algorithms are rules by which the perceptron will update weights (learn) as the user presents data. Backpropagation is the most prevelant method for updating the weights in a multilayer perceptron. This algorithm is a gradient descent method for training the weights in a multilayer perceptron while minimizing the mean squared error between the outputs of the network and the desired outputs (16:50). Wiggins notes that if the learning rate is small enough, the backpropagation algorithm implements a first-order gradient descent search in the weight space for the set of weights which will minimize the sum of the squared errors over the outputs for all exemplars in the data set (32:6). According to several sources, Werbos first derived this technique in 1974. However, it was Rumelhart et al. who first published and popularized the algorithm in their book *Parallel Distributed Processing* (21:54).

Rogers explains that in a multilayer perceptron, the data is introduced to the input layer and propagated through the network in a feedforward manner. Comparing the output of the perceptron with the desired classification yields an error term used to compute a correction for the weights (21:53-56). Listed below is the Backward Error Propagation Algorithm.

1. Initialize weights and thresholds to small random values.

2. Present training input and desired output.

3. Calculate output.

4. Adapt weights and thresholds according to

$$w_{ij}^{+} = w_{ij}^{+} + \eta \delta_j x_i + \alpha(w_{ij}^{-} - w_{ij}^{--}) \tag{2}$$

where w_{ij} is the weight from node i to node j in the next layer, x_i is the output of node i, and δ_j is the *error* associated with node j. η and α are learning rates (for example constants of .35 and .7 respectively). w_{ij}^{+} is the new weight value and w_{ij}^{-} is the old weight value. w_{ij}^{--} is the value of the weight before the last update. Thresholds are adapted similarly where x_i is replaced by "+1" if the threshold is *added* to the weighted sum and "-1" if it is *subtracted*. The δ_j are defined as follows:

$$\delta_j = \begin{array}{ll} y_j(1-y_j)(d_j - y_j) & \text{for output node } j \\ x_j(1-x_j)\sum_k \delta_k w_{jk} & \text{for hidden node } j \end{array} \tag{3}$$

where d_j is the desired output for output node j and y_j is the actual output. For hidden nodes the δ_k are the errors for the layers above.

There are two versions of the correction calculation: instantaneous update and batch update. Instantaneous update examines the gradient of the error surface after the network incorporates each training vector. Batch update examines the gradient after the network sees all the training vectors (22:15).

It is important to note that under conditions with local minima, the back-propagation training algorithm will not necessarily find the best approximation for a given network structure. Rumelhart and McClelland, however, claim that local minima are unlikely to occur in networks with many hidden units. According to these researchers, the added degrees of freedom in such networks by increasing the dimension of the search space, actually increase the likelihood that the search will converge over a convex surface (32:6).

2.1.5 High-Order Inputs. High-order networks refer to networks that have as their inputs second, third, or greater order terms. For example, if two inputs x_1 and x_2 represent two seperate pieces of information, then x_1^2 or $x_1 x_2$ may represent pieces of information more important to discrimination than either term seperately. Adding these high-order terms as inputs to a multilayer perceptron may not decrease the overall error achieved, but training time should decrease.

Giles says the following about high-order networks:

> High-order neural networks have been shown to have impressive computational, storage, and learning capabilities. This performance is because the order or structure of a high-order network can be tailored to the order or structure of a problem. Thus, a neural network designed for a particular class of problems becomes specialized, but also very efficient in solving these problems. (9:4972)

Because of the number of possible inputs to this type of network, Giles suggests several methods of choosing a representative set of terms. These techniques include

1. matching of the order of the network to the order of the problem
2. implementation of invariances if it is known a priori that the problem possesses a given set of invariances
3. calculation of correlations
4. generation of representations adaptively (9:4977- 4978).

The method of calculating correlations seems most appropriate for this research effort. One way to determine which terms will be useful in a high-order network is to calculate correlation matrices for a representative sampling of the mapping to be generated. The entries in the correlation matrix that are largest correspond to the input terms that are most highly correlated with the output and therefore, should be included in the network (9:4977-4978).

2.1.6 Data. Often the input vectors must be normalized in some fashion so that no one feature dominates the classification process. In his dissertation, Ruck states that the following normalization algorithm has been shown to be very effective on many different types of pattern recognition data:

$$x'_{ij} = \frac{x_{ij} - \hat{\mu}_j}{\hat{\sigma}_j} \tag{4}$$

where x_{ij} is the original value of the ith vector's jth feature, x'_{ij} is the normalized value, $\hat{\mu}_j$ is the mean of the jth feature in the training set, and $\hat{\sigma}_j$ is the standard deviation of the jth feature in the training set (22:15). Normalization also can be accomplished by scaling the features in each vector to values between zero and one based on the values of the features in the training set.

2.1.7 Feature Extraction for Multilayer Perceptrons. When employing neural networks of any type, an objective is to limit the number of input features. Devijver cites "the curse of dimensionality" as the primary reason for limiting these features (4:187). The dilemma is that as the number of features increases, the number of training vectors required in the training set also increases. In what has become a rule in the neural network discipline, Foley describes a "reasonable rule of thumb" in determining the number of training vectors required based on the number of features to be input into the network. Accordingly, he states that the greater the ratio of training vectors per class to the feature size is, the better the network's results are. The guide is that the ratio should be greater than three. Satisfying this condition, Foley summarizes, ensures the test set error rate is close to the actual error rate (8:623).

Beyond the importance of limiting the number of features, an aim is to include only those features that make a significant contribution to the network. Ruck's article "Feature Selection Using a Multilayer Perceptron" develops a method for ranking

features called saliency (23). To produce the saliency measure, "the sensitivity of the network's output to its input is used to rank the input feature's usefulness" (23:42).

The calculation of the saliency metric begins with the calculation of the derivative of the output with respect to a specific input. When the sigmoid nonlinearity is used for a network with a single hidden layer, this derivative is

$$\frac{\partial z_i}{\partial x_j} = z_i(1 - z_i) \sum_m w^2_{mi} \delta^1_m w^1_{jm} \tag{5}$$

where z_i is the output of node i in the output layer, w^2_{mi} is the weight connecting the hidden layer with the output layer, w^1_{jm} is the weight connecting the input layer with the hidden layer and $\delta^1_m = x^1_m(1 - x^1_m)$. From the equation above, it is apparent that the derivative depends on the inputs to the network as well as the weights within the network.

Finally, Ruck defines saliency for a feature input j as:

$$\Lambda_j = \sum_{x \in X} \sum_i \sum_{x_j \in D_j} \mid \frac{\partial z_i}{\partial x_j}(\mathrm{x}, \mathrm{w}) \mid \tag{6}$$

where x indicates the multi-dimensional vector inputs, X is the set of all training vectors, w represents the weights in the network, i is the index of elements of x and D_j represents a set of points over which the input x_j will be sampled (23:43). D_j is usually a set of several uniformly spaced points over the expected range of the inputs. The saliency measure ranks the features from most significant to least significant based on the value of the formula above. It is important to note that the weights used in the saliency calculation are fixed weights from a *trained* network.

Devijver describes the use of a probability of error criterion to determine the significant input features. Although this method is more widely known and simpler to implement than saliency, it is unreliable (4:215). The error criterion calculates the probability of error using each feature individually *in succession*. Saliency examines

each feature individually, however, the features are examined arbitrarily close to actual training points (22:32).

A simpler method of determining the relative significance of the input features once the network has been trained has been suggested by Tarr. He states the following:

> When a weight is updated, the network moves the weight a small amount based on the error. Given that a particular feature is relevant to the problem soultion, the weight would be moved in a constant direction until a solution with no error is reached. If the error term is consistent, the direction of the movement of the weight vector, which forms a hyperplane decision boundary, will also be consistent. ...If the error term is not consistent, which can be the case on a single feature out of the input vector, the movement of the weight attached to the node will also be inconsistent. In a similar fashion, if the feature did not contribute to a solution, the weight updates would be random. In other words, useful features would cause the weights to grow, while weights attached to non-salient features would simply fluctuate around zero. (28:44)

Therefore, the following alternate saliency metric is proposed:

$$\tilde{\Lambda}_i = \sum_k w_{ik}^2 \tag{7}$$

Which is simply the sum of the squared weights between the input layer and the first hidden layer.

2.2 Multivariate Statistical Analysis

According to White, "...learning procedures used to train neural networks are inherently statistical techniques" (31:425). Appropriately, the neural network approach to classifier development should be compared to statistically based techniques. This literature review describes certain aspects of multivariate methods as they apply to the two specific applications. Specifically, these aspects include: the

general structure, the method of factor selection, data requirements, the structure of the problem, and two specific statistical techniques.

2.2.1 Multivariate Discriminant Analysis. Dillon and Goldstein define discriminant analysis as "... a statistical technique for classifying individuals or objects into mutually exhaustive groups on the basis of a set of independent variables." Further, the method involves deriving linear combinations of the independent variables that will discriminate between the *a priori* defined groups in such a way that the misclassification rates are minimized (5:360).

Dillon and Goldstein go on to describe discriminant analysis as a rather simple "scoring system" that assigns to each individual in the sample a score that is essentially a weighted average of the individuals's values on the set of independent variables (5:361). Once a "score" is determined, a decision on group membership can be made, or the score can be transformed into a probability of belonging to each of the groups. The individual is then classified into the classification group with the highest probability of membership.

2.2.2 Techniques for Implementation. Two discriminant analysis algorithms will be applied to the problem of determining pilot retention rates to determine a classification rule for each pilot's decision and to the problem of determining a functioning/nonfunctioning API projectile.

K-Nearest-Neighbor Discrimination. Nearest-neighbor discriminant analysis is a nonparametric method for classifying observations into one of several classes on the basis of one or more quantitative variables. This technique uses a large number of correctly classified sample patterns rather than any knowledge of the underlying statistical distribution. As Devijver states, "... the analysis provides a mathematical justification for the assumption that patterns that are close together (in the feature space) are likely to belong to the same pattern class" (4:18).

Letting x_1 and x_2 represent two observation vectors, the k-nearest-neighbor algorithm computes the Mahalanobis distance between x_1 and x_2 based on the total-sample covariance matrix T:

$$d^2(x_1, x_2) = (x_1 - x_2)'T^{-1}(x_1 - x_2) \tag{8}$$

or optionally the Euclidean distance:

$$d^2(x_1, x_2) = (x_1 - x_2)'(x_1 - x_2) \tag{9}$$

Using the nearest-neighbor rule, x_2 is classified into the group corresponding to the point x_1, that yields the smallest $d^2(x_1, x_2)$. Using the k-nearest-neighbor rule, the k smallest distances are saved. Of these k distances, let n_i represent the number of distances that correspond to group i. The posterior probability of membership in group i is:

$$P_i = \frac{n_i prior_i}{\sum n_j prior_j} \tag{10}$$

where j ranges over all classes. Then x_2 is assigned to the group for which P_i is a maximum, unless there is a tie for largest or unless this maximum probability is less than a specified threshold. Normally, k is set to all odd integers between one and nine (odd to ensure a majority).

The value of k which produces the least classification error is actually used. When $k = 1$ is used in the nearest-neighbor rule, x_1 is classified into the group associated with the x_2 point that yields the smallest squared distance $d^2(x_1, y_1)$ (24:560).

The k-nearest-neighbor algorithm requires no assumptions about the distributions of the variables (22:9). In addition, little information is available on the

effects of categorical versus ordinal data in the accuracy of the resulting classification scheme.

Logistic Discrimination. Binary response variables arise in many fields of study. Logistic regression is used when the response is a binary function of a set of factors—as in categorization into one of two classes (24:42). Let x be the vector of factors, G_1 be group 1 and G_2 be group 2. In the case where $P(x \mid G_1)$ and $P(x \mid G_2)$ are multivariate normal with means μ_1 and μ_2, respectively and common covariance matrix, then:

$$P(G_1 \mid \mathbf{x}) = \exp(\alpha + \beta'\mathbf{x})P(G_2 \mid \mathbf{x}) \tag{11}$$

$$P(G_2 \mid \mathbf{x}) = \frac{1}{1 + \exp(\alpha + \beta'\mathbf{x})} \tag{12}$$

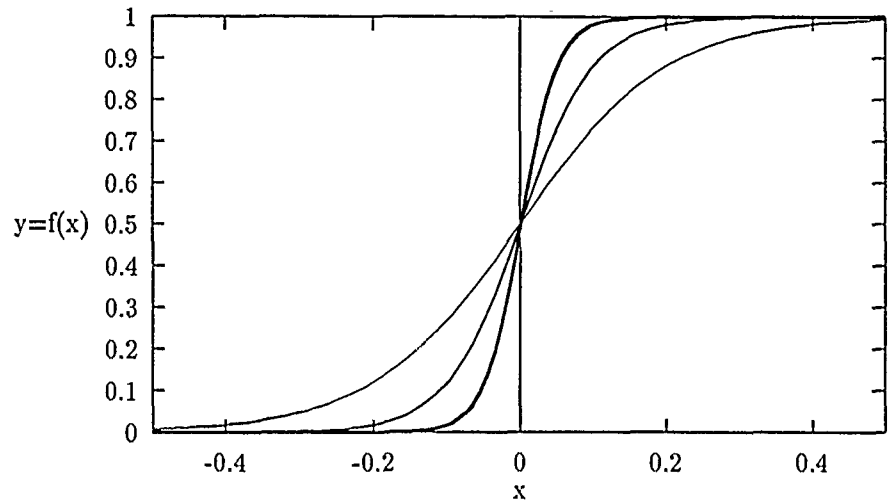
where α and β are the parameters to be found. This equation describes a multivariate logistic function for two groups (5:386). See Figure 4 for an illustration of several variations of this function. The logistic function represents an S-shaped surface with inflection always occurring at the value 1/2 and asymptotes at 0 and 1. Changes in α shift the surface laterally and changes in the β vector affect its dispersion.

A standard method for estimating the parameters α and β is to use a maximum likelihood function. The likelihood equations are nonlinear in α and β. It would be difficult to attempt to find them by hand, but computer packages such as SAS have these options available.

The SAS LOGISTIC procedure uses the Iteratively Reweighted Least Squares (IRLS) Algorithm to compute estimates of the parameters in the model. Let Y_j be the response variable corresponding to the known vector $\mathbf{x}_j'$ of explanatory variables. Consider the multinomial variable $\mathbf{Z}_j = (Z_{1j}, \ldots, Z_{(k+1)j})'$ such that

$$Z_{ij} = \begin{array}{ll} 1 & \text{if } Y_j = i \\ 0 & \text{otherwise.} \end{array} \tag{13}$$

Figure 4. Multivariate Logistic Function



With p_{ij} denoting the probability that the jth observation has response value i, the expected value of $\mathbf{Z}_j$ is $\mathbf{p}_j = (p_{1j}, \ldots, p_{(k+1)j})'$. The covariance matrix of $\mathbf{Z}_j$ is $\mathbf{V}_j$, which is the covariance matrix of a multinomial random variable for one trial with parameter vector $\mathbf{p}_j$. Let γ be the vector of regression parameters; in other words, $\gamma' = (\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_s)'$. And let $\mathbf{D}_j$ be the matrix of partial derivatives of $\mathbf{p}_j$ with respect to γ. The estimating equation for the regression parameters is

$$\sum_j \mathbf{D}_j' \mathbf{W}_j (\mathbf{Z}_j - \mathbf{p}_j) = 0 \tag{14}$$

where $\mathbf{W}_j = w_j \mathbf{V}_j^-$, w_j is the weight of the jth observation, and $\mathbf{V}_j^-$ is a generalized inverse of $\mathbf{V}_j$. LOGISTIC chooses $\mathbf{V}_j^-$ as the inverse of the diagonal matrix with $\mathbf{p}_j$ as the diagonal.

The estimates are obtained iteratively as

$$\hat{\gamma}_{m+1} = \hat{\gamma}_m + (\Sigma_j \hat{\mathbf{D}}_j' \hat{\mathbf{W}}_j \hat{\mathbf{D}}_j)^{-1} \Sigma_j \hat{\mathbf{D}}_j' \hat{\mathbf{W}}_j (\mathbf{Z}_j - \hat{\mathbf{p}}_j) \tag{15}$$

where $\hat{\mathbf{D}}_j$, $\hat{\mathbf{W}}_j$ and $\hat{\mathbf{p}}_j$ are respectively $\mathbf{D}_j$, $\mathbf{W}_j$, and $\mathbf{p}_j$ evaluated at $\hat{\gamma}_m$. The expression after the plus sign is the step size (25:1088).

According to Dillon, "One advantage of using logistic functions for discrimination is their wide applicability." Specifically, the logistic discrimination models can be used in the following situations:

1. Densities are multivariate normal with equal covariance matrices.
2. The measurements are independent Bernoulli variables.
3. The Bernoulli variables follow a loglinear model with equal second- and higher-order effects.
4. Situations (1) and (3) are combined (5:386).

Dillon and Goldstein state that the application of linear regression is appropriate even when categorical and ordinal factors are mixed (5:387). Although the logistic regression method should be used under conditions of multivariate normal densities and equal covariances, this method may also apply under nonnormal conditions and may be robust enough to produce an accurate classifier.

2.2.3 Factor Selection for Multivariate Discriminant Techniques. According to Dillon, when several independent variables are available for inclusion in the discriminator, the common practice is to allow a stepwise selection procedure determine the variables that should be included (5:375-379). These factor selection methods often use single predictor F-values or squared partial correlation as the criteria for selection.

This method begins with no variables in the model. At each step, the variable is entered that contributes most to the discriminatory power of the model by a certain criterion. In the SAS STEPDISC (for stepwise discrimination) procedure, that criterion is the Wilk's lambda (likelihood ratio) criterion (25:1494). In the SAS LOGISTIC procedure, the criterion is the adjusted chi-squared statistic (25:1076).

Variables are entered into and removed from the model in such a way that each forward selection step is followed by one or more backward elimination steps (25:1076). The stepwise procedure stops when it is not possible to enter any more variables and the only variable that can be eliminated is one that just entered.

It is important to realize that in many stepwise selection methods, only one variable can be entered into the model at each step. The selection process does not take into account the relationships between variables that have not yet been selected. Therefore, some important factors could be omitted from the model (25:1495).

Once factors have been determined according to some criteria, there are several methods for identifying which variables contribute the most to the discrimination. Traditionally, there are two methods for determining the importance of a predictor, (1) F-values for each predictor are rank ordered, and (2) standardized discriminant weights are calculated. The problem with these methods is that they may be inaccurate if the predictor variables themselves are highly correlated. Dillon states that an alternate method for finding the contribution of the variables is the calculation of discriminant loadings. A discriminant loading gives the simple correlation of a variable with a discriminant function. "Discriminant loadings have been extensively used in the multiple group discriminant problem for the purpose of labeling the discriminant axes that are uncovered and subsequently retained." (5:372-373)

Dillon says that discriminant loadings reflect common variance among the predictors, whereas standardized discriminant weights examine predictor intercorrelations and may, therefore, be unstable (5:373). Subsequently, discriminant loadings can be useful tools. To calculate discriminant loadings, the correlation of the predictors with the discriminant function is calculated i.e., $corr(\mathrm{x}, \mathrm{b}'\mathrm{x})$ (2):

$$corr(\mathrm{x}, \mathrm{b}'\mathrm{x}) = (\mathrm{b}'cov(\mathrm{x},\mathrm{x})\mathrm{b})^{-\frac{1}{2}} corr(\mathrm{x},\mathrm{x}) D_{\mathrm{x}}^{-\frac{1}{2}} \mathrm{b} \qquad (16)$$

where x is the vector of predictors (features), b'x is the discriminant function resulting from the features and the estimated coefficients and D_x is the matrix of diagonal entries of $cov(x, x)$.

2.3 Comparing Classification Methods

Due to the great differences in the development and implementation of the methodologies to be studied here, the development of a means to compare the methods is critical. Weiss reports on results of an extensive comparison of classification methods on the same data sets. Weiss concludes that the choice of a classification model is highly dependent on the problem itself (18:14). For the specific problems of predicting retention rates and classifying API projectiles, the error rate and computational complexity will be the deciding factors. The following sections discuss methods for comparing these factors for the classifiers to be examined.

2.3.1 Error Rate Estimation. Although the discovery of the discriminant function and optimal neural net structure are important, equally important is the estimation of the classification performance of discriminator. Several sources suggest techniques for estimating error rates (2:353-366; 9:723- 741; 4:316-317). Each error estimation technique has positive and negative attributes associated with it in the context of estimating retention rates for pilots. The following discussion outlines three of these methods.

Resubstitution is a method of estimating the actual error by using the same set of samples to design a classifier and test it. Devijver states that this method "is uniformly poor." (4:346) This method has the disadvantage of underestimating the error and may yield misleading results. On the other hand, Foley points out that if the ratio of samples per class to dimensions is greater than three, then the error rate on the training set is a good predictor of the error rate when the classifier is used on the test set (8).

An obvious alternative to resubstitution is a *holdout* estimation of the error. This method partitions the data into two mutually exclusive subsets and uses one subset to train the classifier and the other subset to test the classifier. In contrast to the resubstitution method, this method may overestimate the actual error. In addition, this method does not make good use of the available data. Normally, when using this method, the classifier will be redesigned using all the data after an error estimate has been made.

In a discussion by Weiss concerning the merits of several error estimation techniques, he states

> The simplest technique for "honestly" estimating error rates, the holdout or H method, is a single train and test experiment. This technique breaks the sample cases into two groups: a training group and a test group. The classifier independently derives the error rate from the training cases, and the error estimate is the performance of the classifier on the test cases. (29:3)

A final method of estimating error is the *rotation estimate* or *crossvalidation.* For this technique, the training sample is partitioned into k subsets. All but one subset is used to develop the classification rule and the rule is tested on the remaining subset. The left-out subset is returned to the design set and the process is repeated k times. Provided enough data is available, this method reduces some of the bias inherent in the resubstitution method. In many cases, however, the mean square error may be large (11:37).

2.3.2 Comparing Computational Complexity. The following statement by Wiggins summarizes the complexity of comparing classification methodologies.

> The complexity of neural network models makes then more difficult to interpret than standard parametric models. Even if the model performs well in- and out-of-sample, the reason for its performance and its behavior

> over different input ranges cannot be evaluated directly. The very aspect of neural networks that gives them powerful analytic capability makes them rather difficult to interpret. (32:36)

In addition, due to the available resources, the classifiers developed in this research effort were developed on completely different computers and with completely different software. Any comparisons done on the complexity of the methods must be subjective.

2.4 Determining Personnel Retention Rates

In-depth analysis of the literature shows no evidence of the use of multilayer perceptrons or discriminant analysis to predict retention rates for Air Force pilots. However, authors have proposed several other methods to predict the "stay" or "leave" decision of rated Air Force personnel. Guzowski used economically quantifiable variables in a linear regression setting to predict the retention rates of Air Force pilots (13:vi). Simpson also used economically motivated variables in a more sophisticated regression analysis (27:vii). Both authors stress the importance of economic indicators in an analysis of this type.

Analysts in the Navy used similar methods to predict the career decisions of military officers. Both Whalen and Shigley analyzed factors affecting the retention of officers in the medical profession using logistic regression (30, 26). Validation of these analyses reveals the weaknesses of regression models for accurate prediction. Both studies emphasize the need for increased sophistication in the prediction process.

Cromer and Julicher developed a model to descirbe Air Force pilot retention rates. Their objective included building a model based on economic conditions to determine the significance of airline hires on pilot retention. The methods used by Cromer and Julicher included factor analysis, stepwise multiple regression and multiple regression with lagged retention rates. Their results showed that no model accurately described retention rates (27:9).

A model developed by Gotz and McCall of RAND Corporation calculates the probability that an Air Force officer will voluntarily remain in the service based on a given set of retirement, compensation, and promotion policies. The voluntary retention rates are determined in the stochastic dynamic program by finding the individual officer's optimum time to leave the military. According to Gotz, the optimum time occurs when the individual's expected present value of pecuniary and non-pecuniary returns are maximized (10).

2.5 Neural Networks in Personnel Analysis

Neural Networks have been used to model several aspects of the Air Force personnel system. An extensive review of the neural network literature by Wiggins indicates that these networks have "proven superior to more traditional analytic techniques in many applications.(32:i)" Wiggins discusses the use of neural networks to determine the enlistment behavior of high school seniors. Each potential enlistee was classified as either a likely enlister or non-enlister based on a set of individual characteristics, current status and expectations (32:24).

Retention and reenlistment of enlisted airmen is one of the most heavily researched areas in the Air Force personnel system. Researchers at Air Force Human Resources Laboratory (AFHRL) have investigated the use of neural networks to attempt to explain and quantify the factors which affect reenlistment decisions made by individual airmen. Wiggins states that "This is an archetypical classification problem and one to which neural networks are particularly suited." (32:24)

2.6 Classifying the Performance of API Projectiles

> An armor piercing incendiary (API) projectile is a bullet that is capable of perforating light armor and consists of a flammable mixture the is generally encased in the nose of the projectile body. The design of the projectile allows the jacket over the nose to deform or peel off upon

impact of the target skin. The incendiary mixture will flash as the bullet continues its flight. (15)

Incendiary functioning (IF) is dependent upon the interaction of forces that occur between the projectile and target material. At present, there are six classifications for incendiary functioning: complete, partial, slowburn, delayed, non-functioning, and frontal. Pettit provides complete definitions for these six classes (19).

Research findings concerning projectile IF did not really begin until the mid 1970's. The vulnerability community recognizes Mayerhofer's work, in 1974, as the cornerstone of IF prediction methods. Mayerhofer's goal was to find

> tools for the analyst that will first, enable him to predict the type of function that will occur for a particular set of conditions, and secondly, to predict flash delay time, duration, location, and size associated with that type of function.(17:22)

The prediction method suggested by Mayerhofer was based on projectile force and impulse (17:22-23).

$$Force = \sigma_S \pi D t \sec\theta \tag{17}$$

$$Impulse = MV \tag{18}$$

where

- σ_S = ultimate shear stress of target (psi)
- D = projectile diameter (in)
- t = target thickness (in)
- θ = obliquity angle (radians)
- M = projectile mass (lbs/g)

- V = impact velocity (fps)

In 1977, Falcon Research and Development (FRD), presented methods for predicting fires once an API projectile perforated a fuel tank. The correct prediction of an API projectile's IF constituted a critical portion of the method. Using Mayerhofer's method, FRD altered his method to establish their own IF prediction model (7:22) Their work would be the basis for the IF prediction methods used today. The FRD equations require numerous target material properties. Due to the use of composite materials, it was necessary to substitute the properties of aluminum into these equations. This proved to cause a large prediction error rate.

Reynolds, in his thesis entitled *A Response Surface Model for the Incendiary Functioning Characteristics of Soviet API Projectiles Impacting Graphite Epoxy Composite Panels*, applied linear regression and multivariate analysis to 118 test shots to create prediction equations for projectile IF into composite material (20). Using traditional discriminant analysis techniques, he classified three types of IF. Since the work accomplished by Reynolds, the Survivability Enhancement Branch has increased the number of shots in their database to 281.

A thorough examination of the pertinent literature does not reveal the use of neural networks to attempt to classify the performance of API projectiles.

III. Methodology

This chapter provides a discussion of the methodology used to examine each of the following research areas

- Development of Applicable Decision Factors
- Data Collection and Orientation
- Development of Classification Methodology
- Feature Selection
- Comparisons with other Classifiers
- Applications to Specific Problems

3.1 Determining Applicable Data Elements

For a given application, an infinite number of features could be considered relevant in the discrimination of groups of data. In order to apply any multivariate technique, it is necessary to reduce the number of features to a finite set. An initial consideration should be the availability of data. A certain piece of data may be especially germane to classification, however, not be available in any useful form. For example, for the problem of classifying pilots, information concerning the individual's next assignment would be especially useful, however, this information is unavailable until the assignment is actually made.

Most often, a subject matter expert may be in the best position to determine which input features will be most applicable and which are meaningless. However, it is only after the analyst has actually examined the input features and attempted to construct the discriminator, that a judgement can be made as to whether the features actually separate the data into groups.

3.2 Data Collection and Orientation

Application of a pattern classifier requires selection of features that must be tailored for each problem domain or application. Features should contain information required to distinguish between classes, be insensitive to superfluous variability in the input, and also be limited in number to reduce the number of training vectors required. Often the analyst must use subjective judgement to select and configure the information that will best discriminate between classes.

Once a finite set of input features has been selected, it is necessary to translate the data into a format appropriate for input to the classifier. If the data is categorical, it is necessary to decompose these categories into binary variables, or groups of binary variables. Continuous data requires no translation of this type. For the two group problem, one group's classification was translated to the value "0" and the other class to the variable "1". (The actual numeric value of the membership variable may be different for different types of classifiers.)

It was stated above that the data elements are translated into "binary" elements. In the strictest sense, binary features were not used. Traditionally, binary variables take on the value "1" when something is "true" or considered "on" and take the value "0" when "false" or "off." Due to the structure of the backpropagation algorithm used by the multilayer perceptron, "1" and "-1" were used. As discussed earlier, the backpropagation algorithm updates the multilayer perceptron's weights based on the difference between the desired output and the actual output of the network. If binary features "0" and "1" were used, the zeros input to the perceptron would have no effect on the weight updates. Values of "-1", on the other hand, will cause the network to update its weights. Note that the data element which represents the desired output is still coded as "0" or "1" since the multilayer perceptron's outputs will be between "0" and "1" when a sigmoidal nonlinearity is used.

After the translation of all data elements, a FORTRAN program (shown at Appendix A) randomly assigned each feature vector from the population into one of three sets:

- The Training Set: This set of feature vectors was presented to the multilayer perceptron and the discriminant analysis classifiers for training. When each feature vector is introduced to the specific classifier, the vector will be appended with the actual decision of the individual, i.e., the desired output of the classifier.
- The Test Set: For the multilayer perceptron technique, the test set is used to test the accuarcy of training while training is ongoing. After each epoch (i.e., each complete presentation of the training set), the test set vector was presented to the network, classified, the classification compared to the desired classification, and a classification error computed. These test vectors act as controls for determining when the accuracy of the perceptron is at an acceptable level. For the discriminant analysis techniques, however, the test set and the training set was combined and presented to the classifiers for training.
- The Validation Set: For all classification techniques, the validation set was used to estimate an error rate for classification accuracy. After the classifiers were optimally trained, this set was presented to the classifiers. Feature vectors were classified and that classification was compared to the true classification of the vector.

3.3 Development of Classification Methodology

The data sets for each application were used independently to determine the specific multilayer perceptron structure and specific multivariate techniques that were optimal. The overall methodology employed in developing each of the classifiers is as follows:

1. Use the training set and the test set to train the classifier to classify objects onto Group 1 or Group 2 (since each of the applications was a two-group problem). In other words, given the feature vectors in the training and test sets, determine the optimal weights or coefficients that produce the given, desired output for some input. Note that "optimal" here, means optimal in terms of the particular method under consideration—multilayer perceptron or multivariate discriminate.
2. Iterate through the various forms or structures of the discriminators to arrive at a discriminator that yields the best classification accuracy for the training and test sets. In the case of the multilayer perceptron, this included varying the number of middle nodes, changing the learning rate and the momentum, etc. In the case of the multivariate discriminant analysis techniques, these iterations could include varying the significance levels, changing a distance measure, or changing the variable selection criteria for the validation data.
3. Given a trained, minimum error classifier, introduce the feature vectors from the validation set and determine the classification of each of the the vectors. Next, compare the classification of all vectors in the validation set to their actual classification and determine an estimator for the error rate of the classifier.
4. Repeat the steps above for another training set, test set, and validation set randomly selected from the population.

Below, the methodology for the development of each of the classification techniques is discussed in more detail.

3.3.1 Investigation of Multilayer Perceptron Techniques The method for training a multilayer perceptron is illustrated in Figure 5. A FORTRAN program implements the training of a multilayer perceptron for the two specific applications. The program allows the user to vary certain parameters controlling the structure of the perceptron which affect the accuracy and speed of the training. The follow-

ing paragraphs describe both the FORTRAN program and certain implementation decisions.

3.3.2 The Multilayer Perceptron Program. To implement the backpropagation algorithm, the computer code for a single hidden layer multilayer perceptron was written in FORTRAN 77. This multilayer perceptron uses a sigmoid as the nonlinear transformation. The code itself is shown at Appendix B. Several key points concerning the program subroutines are mentioned below.

First, subroutine INPUT takes the feature vectors and desired output from as external file. This external file actually contains the binary transformation of the data elements described above. Data is read into two arrays—one containing the training set feature vectors and the other containing the test set feature vectors.

Once the data is read into these arrays, each continuous feature is normalized to a value between zero and one in the subroutine NORMAL. This normalization is performed to ensure that no one feature dominates the training process. Standardization using the sample mean and sample variance of the features is often suggested for the backpropagation algorithm. This technique was used when possible, however, the technique was not appropriate for binary factors. Binary variables remain coded as "1" and "-1."

The artificial neural net subroutine (ANN) begins by initializing the weights connecting both the input layer and hidden layer and the hidden layer and output layer. Next, the confusion matrices are initialized. In the case of an n class problem, a confusion matrix is an $n \times n$ matrix whose i, j elements are the number of vectors whose desired outputs are group i and are classified by the network as belonging in group j. The rows of the table relate to the actual group membership and the columns give the predicted group membership. Therefore, correct classifications appear on the main diagonal and incorrect classifications appear off the diagonal.

Figure 5. Training Procedure for Multilayer Perceptron

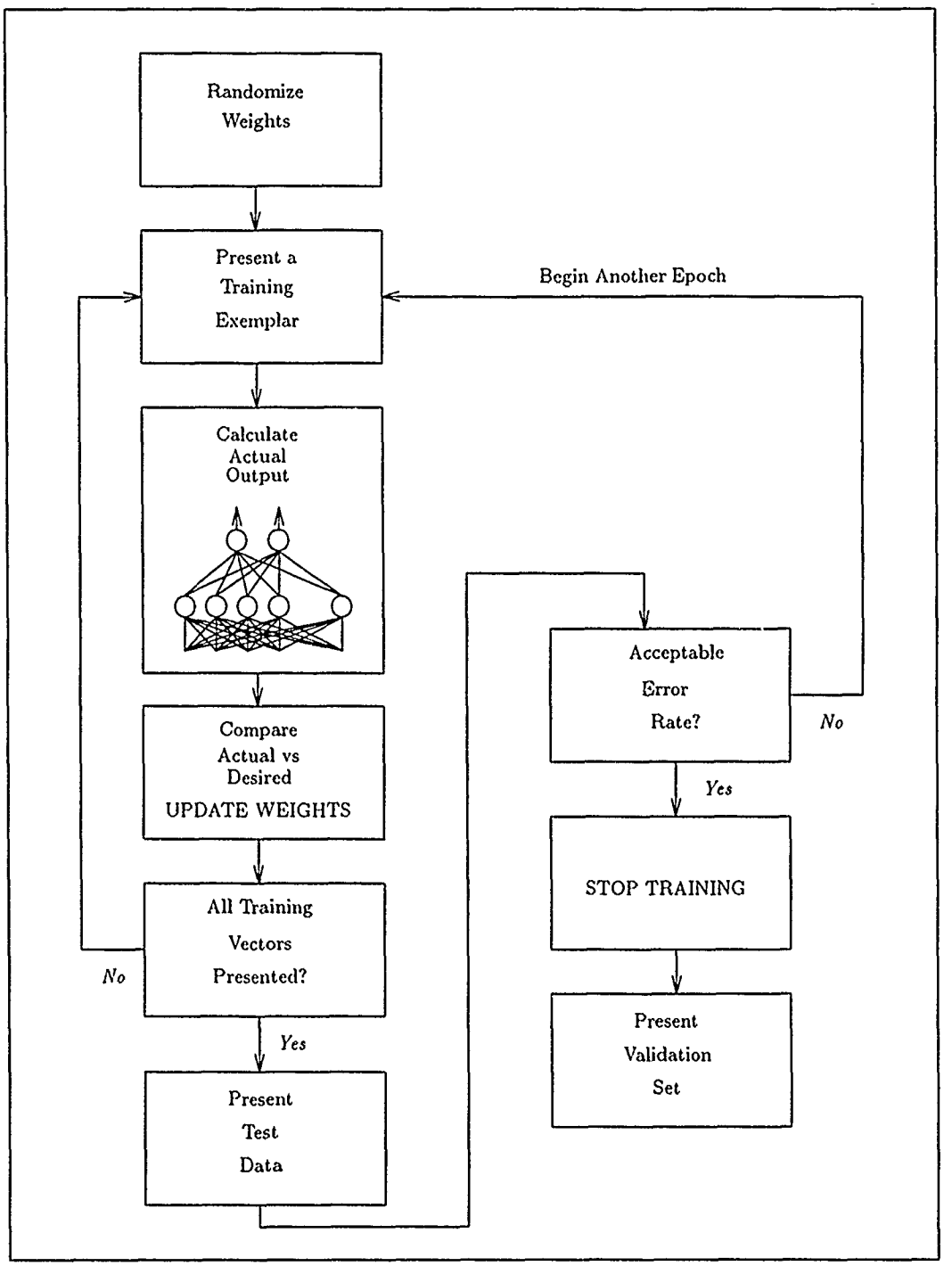
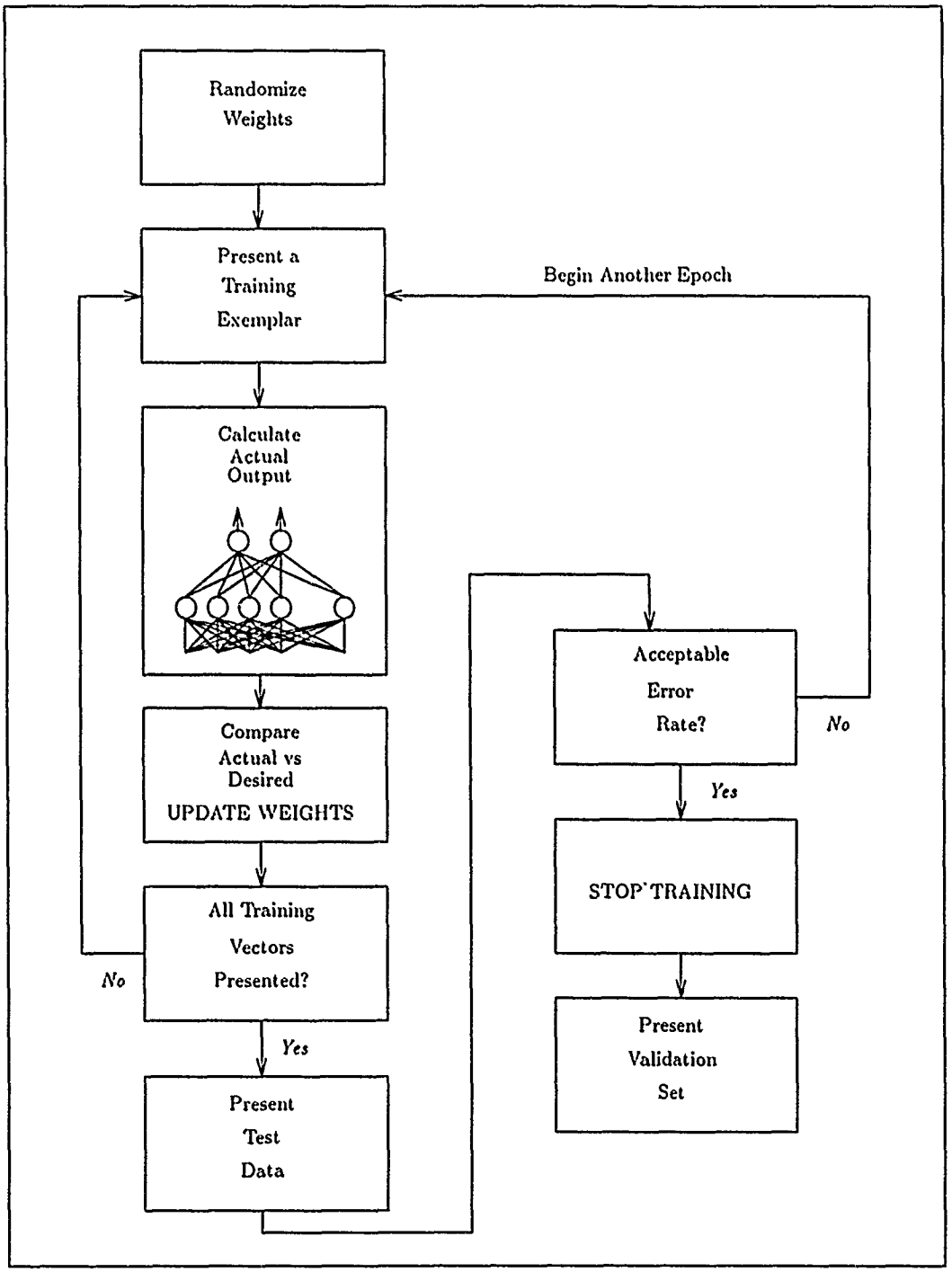


Figure 5. Training Procedure for Multilayer Perceptron



The next phase in the ANN subroutine is the reordering of the feature vectors in the training set and the test set into a random list. Random ordering prevents the network from learning the order of the data and may speed the training time.

The training of the weights in the network is accomplished in the next phase of the ANN subroutine. First, the activation of the hidden layer and output layers are calculated. The activations of the output layer are compared to the desired (known) output and the appropriate weights are updated. After all the training vectors have been presented to the network, the weights are held constant and the test vectors are presented to the perceptron. The error rate of the test set (in conjunction with the error rate of the training set) was used as an indicator of the performance of the network. Each presentation of the set of taining vectors is defined as an "epoch." By collecting the error rate for the entire test set at the end of each training epoch, an error curve can be constructed and a minimum erro: observed somewwhere along this curve. To aid in the analysis of the resulting multilayer perceptron, data pertaining to weights and errors as well as a final confusion matrix is written to several output files.

3.3.3 Structure of the Perceptron. As the analyst applies the multilayer perceptron to a classification problem, several tactical decisions are necessary which affect the resulting structure of the perceptron. Listed below are the major decisions affecting the structure of a multilayer perceptron. Besides the structure of the network, a primary consideration is the form the input features will take. Therefore, a discussion of the use of high-order inputs is also included here.

Implementation Decisions. Initially, when training the multilayer perceptron, the weights must be initialized as random numbers according to some distribution. Questions arise as to the distribution and paramaters of the distribution to be employed. Although the final results should not be affected by the manner in which

the weights are initialized, convergence may be quicker and local minima may be avoided for some distributions.

A second tactical decision that affects the structure of a perceptron is the values of the learning rate (α) and momentum factor (η) in the weight update equations. The learning rate controls the amount of correction to the weights that is made based on the difference between the actual output of the neural network and the desired output. The momentum term causes the weight changes to be modified by some factor of the amount they were modified on the last update. For example, if the weight change on the last iteration was in a negative direction (that is, the weight was reduced) then the momentum term would be negative on this iteration and would cause some proportional decrease of the current weights. The values of α and η were varied between 0.1 and 0.9 for this analysis.

Another decision that affects the structure of a perceptron is the number of nodes in the hidden layer. One major drawback of backpropagation classifiers may be their long training times for certain data sets. Training times are longer when many hidden nodes are used and when the underlying decision regions are especially complex (29). As with other classifiers, training time is reduced and performance is improved if the network is large enough to solve a problem yet not so large as to estimate too many training parameters without the required number of training vectors. For the specific applications examined, the number of nodes in the hidden layer was varied between 5 and 100 nodes.

One of the most important decisions in the application of the multilayer perceptron is when to stop training. When training a multilayer perceptron to discriminate, it is expected that two types of error should decrease as the number of epochs increases. First, the difference between the desired output of the output nodes and the actual output of these nodes should decrease. In other words, if the desired classification was Group 1, then the output of Node 1 should be as close to "1" as possible and the output of Node 2 should be as close to "0" as possible. As the

number of times the data is presented to the network increases, the average amount that each node is different from "1" or "0" should decrease. The average amount that the desired output if different from the actual output for each of the output nodes is called the output error. Both the training set and the test set error rates should decrease. Second, the *number* of feature vectors classified incorrectly in both the test and training sets should decrease as the number of epochs increases.

The goal is to cease training at the point corresponding to a minimum error on the test set. The choice of this point may be difficult since it is necessary to consider the two types of error. Let N_0 be the epoch at which the test set reaches its minimum output error. It is not always a simple task to determine N_0. The error curve may oscillate around some average minimum and even increase as the number of epochs is increased. Consequently, it may not be possible to find the exact value of N_0. For this research, in order to judge the epoch at which the error curve first reaches its minimum, the multilayer perceptron was trained for N epochs, where $N \gg$ than an estimated value of N_0. If the multilayer perceptron is trained for N epochs, and it appears as though the error is continuing to decrease, then N should be increased until a miminum error is observed. Figures 6 and 7 show an examples of output error curves and classification error curves for the training set and test set. In this case, the output error for the test set indicates that $N_0 \approx 400$, however, the classification error does not stabilize until approximately 800 epochs. Hence, the selection of where to stop training a multilayer perceptron is often dependent on whether the application requires accurate output or accurate classification. The decision is most often left to the analyst.

The question arose as to which set of weights to use for classification when several independent training sessions are involved. It was determined that there is no specific relationship between the corresponding weights for each run. An "average" of these weights, for example, would have no meaning. Therefore, the network with the minimum classification error on the test set was used for classification.

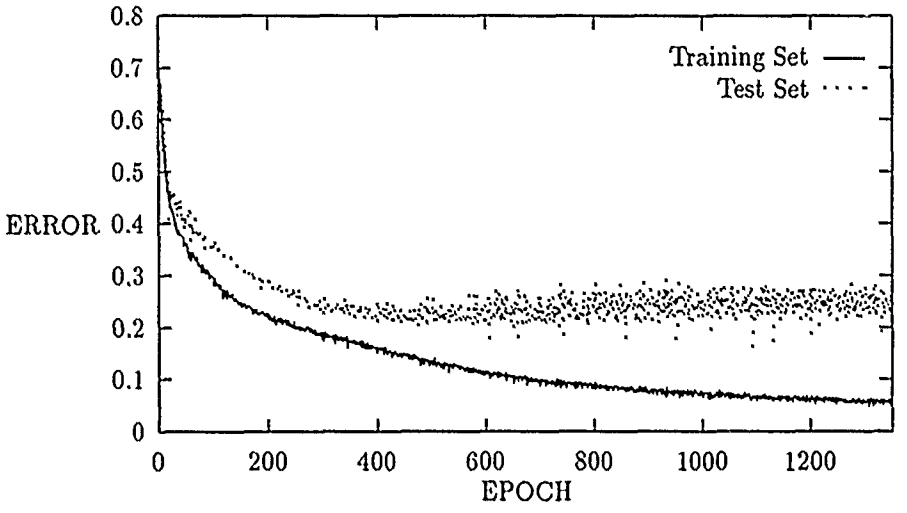


Figure 6. Sample Output Error Curves for Training and Test Sets

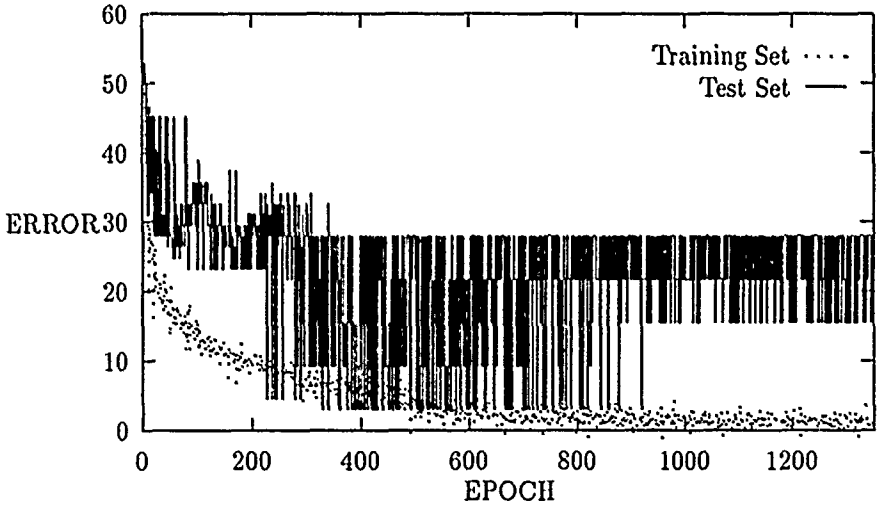


Figure 7. Sample Classification Error Curves for Training and Test Sets

Figure 8 summarizes the procedure used to iterate through the possible multilayer perceptron parameters discussed above and arrive at an optimal network. Initially, the number of middle nodes, the learning rate and the momentum rate were set to values that are suggested by those who use these networks often (21). The network was trained and the number of epochs increased until the minimum test set error is observed. The number of middle nodes continued to increase as long as the minimum error continued to decrease, or the training time was shorter. Then, the number of middle nodes was fixed and the learning and momentum rates were tested over some range. This range of learning and momentum rates depended on the order of the application (i.e., how many feature inputs/feature vectors were involved) and the observed behavior of the error rate as these rates were changed. After the multilayer perceptron was tested over the entire range of learning rates and momentum rates, these parameters were fixed.

This is not an optimal testing methodology since the interactions of the parameters are confounded. One technique to arrive at a true "optimal" structure would be to train the multilayer perceptron for all possible combinations of numbers of middle nodes, learning rates and momentum rates. Due to time constraints, this was not practical.

High-Order Inputs. In order to determine if high-order inputs would produce a more efficient classsifier, products of features were formed and used as inputs. The rationale for using high-order inputs and developing a classifier based on these inputs can best be shown through a two-dimensional example. The "parity problem" (for high-order problems) or "XOR problem" (for two-dimensions) is often used to test classifiers to determine their ability to classify non-linearly separable decision regions. Training a perceptron to perform this separation may require thousands of iterations of the fastest learning rules (9:4973). Figure 9 illustrates the problem.

Rogers describes the problem as:

Figure 8. Procedure for Determining Multilayer Perceptron Structure

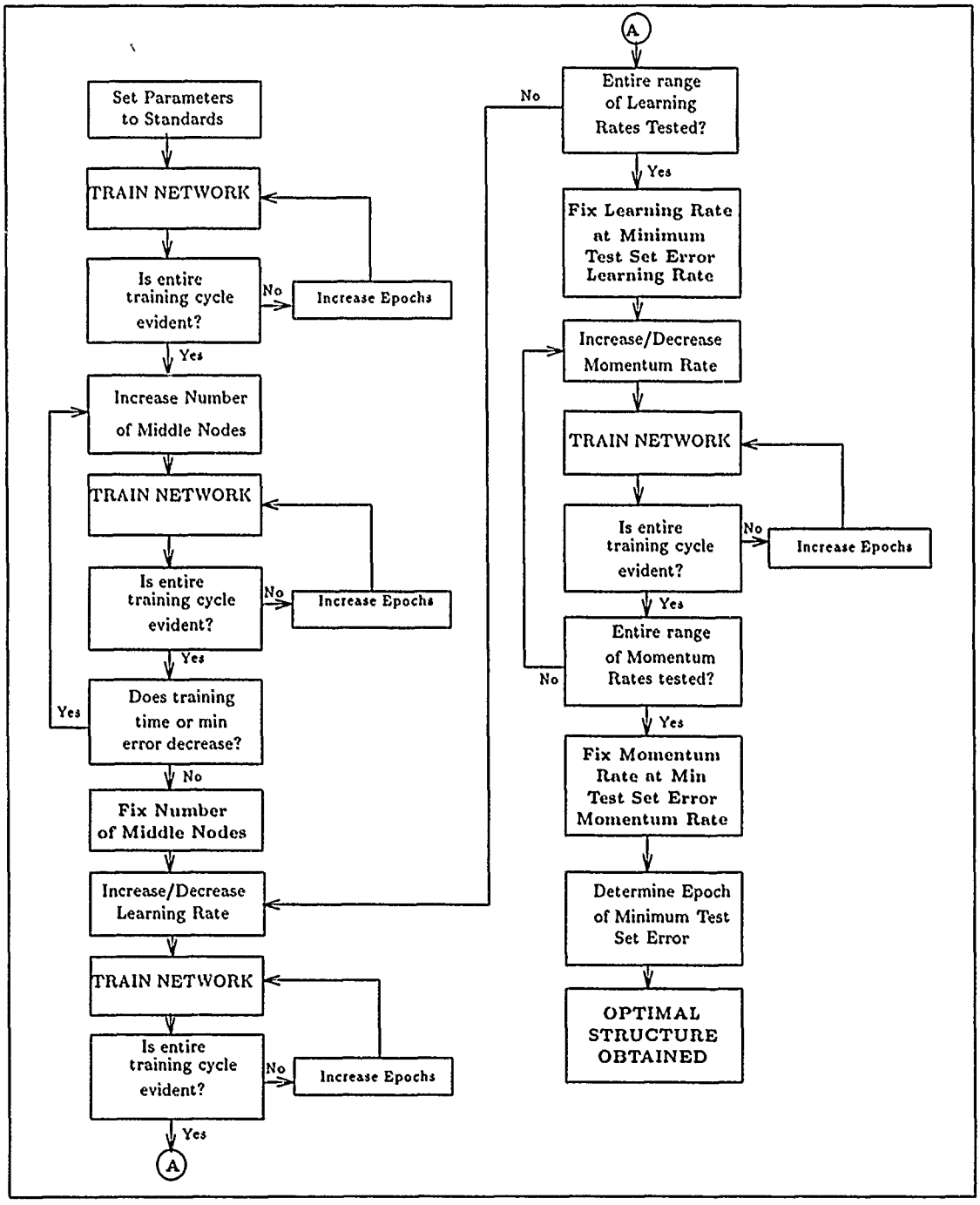
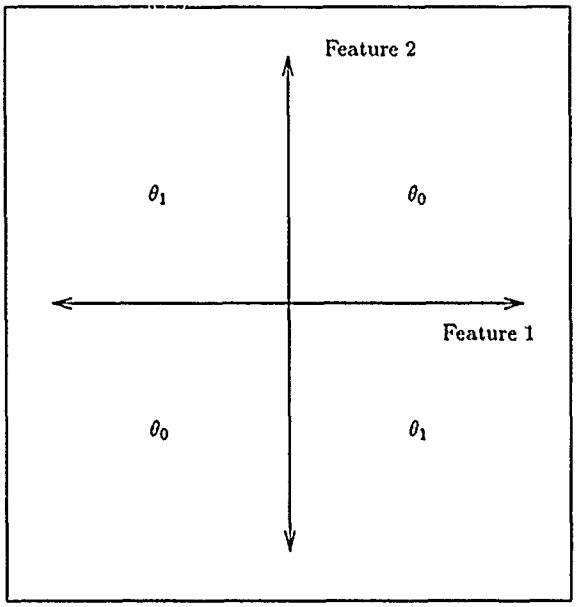


Figure 9. The XOR Problem



> If the features have different polarity (one positive and one negative), then the output should be a one (θ_1), and if they have the same polarity (both positive or both negative), then the output should have a value of zero (θ_0). (21:53)

Note that no single line can be drawn to separate the θ_1 and θ_0 regions. However, if Feature 1 is multiplied by Feature 2, a classification rule for the resulting one dimensional space can easily be developed. The resulting rule would classify a feature vector into class θ_1 if the product of inputs one and two were less than zero and into class θ_0 if the product of the inputs was greater than zero.

In order to formally determine if a high-order network produces a lower minimum error rate than a network with the first-order original inputs, it would be necessary to successively add all higher order terms. However, it has been suggested that an examination of correlation matrices for a representative sampling of the mapping to be generated may prove useful. Giles presents a method to claculate these

correlation matrices as follows:

$$W_2(i,j,k) = \sum_{s=1}^{Np}[y^s(i) - \mathrm{y}(i)][x^s(j) - \mathrm{x}(j)][x^s(k) - \mathrm{x}(k)] \tag{19}$$

where,

$$\mathrm{y}(i) = [\sum_{s=1}^{Np} y^s(i)]/Np \tag{20}$$

and,

$$\mathrm{x}(i) = [\sum_{s=1}^{Np} x^s(i)]/Np \tag{21}$$

$W_2(i,j,k)$ is an element of the second-order (denoted by the subscript 2) correlation matrix for the ith output node correlated with the product of the jth input and the kth input. $y^s(i)$ is the ouput for node i for the sth training exemplar and $x^s(j)$ is the input for node j for the sth training exemplar. In the equations above, Np denotes the number of vectors in the training set. The training set is denoted by $\{(x^s, y^s)|s \in (1, Np)\}$, and y and x are the averages of y^s and x^s over the training set (9:4973). The entries in the correlation matrix that are greatest in absolute value correspond to terms that are highly correlated with the output of the perceptron. According to Giles, these terms "are most likely to make an important contribution to the network when the map is implemented (9:4978).

The correlation matrix in the equation above represents second-order correlations only. In order to investigate third-order and higher correlations it would be necessary to construct larger correlation arrays.

To implement this method of generating high-order representations, a subroutine was included in the ANN program called CORRELATE. This subroutine calculates the second-order correlation matrices according to the equation above, sorts the correlations in terms of their absolute value, and writes the results to a file. The results were interpreted as possible second-order terms to use as inputs to the perceptron. Once again, the impetus to employ high-order inputs, was to exploit

the use of binary features as illustrated by the XOR problem and possibly produce a superior classifier.

Once the highly correlated second-order terms were identified, these terms were included as inputs. The multilayer perceptron with the second-order inputs was trained again to see if the minimum error of the test set decreased or if the training time was less than the original feature inputs.

3.3.4 Investigation of Multivariate Techniques-Logistic Regression. Logistic discrimination is appropriate when both qualitative and quantatative variables are being used to attempt to discriminate between two groups. For the two group classification problem, the SAS LOGISTIC procedure fit a linear logistic regression model for this binary response data by the method of maximum likelihood.

A sample SAS logistic regression program is shown in Appendix D. To implement the LOGISTIC procedure, one need only specify the model to be fit in terms of the independent and dependent variables. In order to allow for easier comparison between the multilayer perceptron technique and the discriminant analysis techniques, the same data format was used for the logistic regression procedure as was used for the perceptron above. That is, when variables are referred to as binary, this implies that they are "1" or "-1." Before estimation begins, the LOGISTIC procedure calculates the global score statistic for testing the joint significance of all explanatory variables in the model. The Maximum Likelihood Estimators (MLEs) of the regression parameters are computed using the Iteratively Reweighted Least Squares (IRLS) algorithm. The estimated covariance matrix of the MLEs is obtained by inverting the expected value of the hessian matrix for the last iteration or the IRLS algorithm.

Within the LOGISTIC procedure, subsets of explanatory variables may be chosen by various model selection procedures. SAS allows for four model-selection methods. The default method fits the complete model as specified in the model

statement. The other three methods are FORWARD for forward selection, BACKWARD for backward elimination and STEPWISE for stepwise selection. In this analysis, the model selection methods used were the default and stepwise selection. To determine which variables should enter and leave the model, the procedure calculates the adjusted chi-squared statistics for all variables and examines the largest of these statistics. If the variable is significant at the level specified, the variable with the largest adjusted chi-squared statistic is entered into the model. The stepwise procedure follows this forward selection step with one or more backward elimination steps until no further variables can be added to the model, or if the variable just added is the only variable that can be removed (25).

To explain further, suppose there are s explanatory variables of interest. The full model has parameter vector

$$\gamma = (\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_s)' \tag{22}$$

where $\alpha_1, \ldots, \alpha_k$ are intercept parameters, and $\beta_1, \ldots, \beta_s$ are slope parameters for the explanatory variables. For a reduced model with t explanatory variables ($t < s$), let $\hat{\alpha}_1, \ldots, \hat{\alpha}_k$ be the MLEs of the unknown intercept parameters for this model, and let $\hat{\beta}_1, \ldots, \hat{\beta}_t$ be the MLEs of the unknown slope parameters for this model. The residual chi-square is the chi-squared score statistic evaluated at γ_0, which is given by

$$\gamma_0 = (\hat{\alpha}_1, \ldots, \hat{\alpha}_k, \hat{\beta}_1, \ldots, \hat{\beta}_s, 0, \ldots, 0)' \tag{23}$$

. The residual chi-square has an asymptotic chi-squared distribution with $s - t$ degrees of freedom. The significance of a specific variable adjusted for the variables already in the model can be determined by comparing the residual chi-square with a chi-square distribution with one degree of freedom (25).

3.3.5 Investigation of Multivariate Techniques–Nearest Neighbor. In SAS, nonparametric discriminant methods are based on nonparametric estimates of group-specific probability densities. Either a kernal method or the k-nearest neighbor method can be used to generate a nonparametric density estimate in each group and to produce a classification criterion. In this analysis, the k-nearest neighbor method with Mahalanobis was utilized. 'The procedure finds the radius $r_k(\mathbf{x})$ that is the distance from x to the kth nearest training point in the metric $\mathbf{V}_t^{-1}$ where $\mathbf{V}_t$ is the pooled within-group covariance matrix(25:683). A sample SAS k-nearest -neighbor program is shown at Appendix E.

Using the k-nearest neighbor rule, the k smallest distances are saved. Of these k distances, let k_t represent the number of distances associated with group t. Then, the estimated group t density at x is:

$$f_t(\mathbf{x}) = \frac{k_t}{n_t v_k(\mathbf{x})} \tag{24}$$

where $v_k(\mathbf{x})$ is the volume of the ellipsoid bounded by $\{\mathbf{z}|(\mathbf{z}-\mathbf{x})'\mathbf{V}_t^{-1}(\mathbf{z}-\mathbf{x}) = r_k^2(\mathbf{x})\}$. When $k = 1$ is used in the nearest-neighbor rule, x is classified into the same group as y where y yields the smallest squared distance $d_t^2(\mathbf{x},\mathbf{y})$(25:680-685). In general, the value of k has an effect on the degree of irregularity in the estimate of the density function—small values of k produce jagged density estimates and large values of k may tend to produce smoother density estimates. Formal methods of determining the best value of k tend to be complicated and problem dependent (25). Accordingly, values of k between one and nine were tested and the value of k that produced the smallest estimated error for the validation set was used.

3.4 Analysis of Input Features

3.4.1 Multilayer Perceptron Input Features When designing a classifier, the features that provide information should be included and those that provide little

information should not be provided as inputs to the network. Once the best multilayer perceptron structure was determined, an analysis of the input features was conducted. The optimal technique for feature selection requires implicit examination of every possible subset of the set of features under consideration (4:207-214). In a situation with very many variables, this would be impractical. The saliency metric allows for the examination of the inputs as they relate to the output values of the multilayer perceptron without examining every subset.

The saliency metric (Λ_j) for feature j when a sigmoid nonlinearity is used is restated below.

$$\Lambda_j = \sum_{x \in X} \sum_{i} \sum_{x_j \in D_j} \left| \frac{\partial z_i}{\partial x_j}(\mathrm{x}, \mathrm{w}) \right| \tag{25}$$

where x indicates the multidimensional vector inputs, X is the set of all training vectors, w represents the weights in the network, i is the index of elements of x and D_j represents a set of points over which the input x_j will be sampled (23:43). D_j is usually a set of several uniformly spaced points over the expected range of the inputs.

The derivative of the outputs with respect to the input can be written as a function of only the weights and activations as follows:

$$\frac{\partial z_i}{\partial x_j} = z_i(1 - z_i) \sum_m w^2_{mi} x^1_m (1 - x^1_m) w^1_{jm} \tag{26}$$

where, z_i is the output of node i in the output layer, x^1_m is the output of node m in the first layer and w^1_{jm} is the weight connecting node j in the input layer to node m in the first layer. The derivation equation above is applicable for perceptrons with a single hidden layer. As the number of hidden layers increases, the calculation of this saliency metric becomes more complex.

As mentioned earlier, another method for determining which inputs are significant is to examine the weights connecting the inputs and the first hidden layer. The equation for this saliencey metric is restated below.

$$\tilde{\Lambda}_i = \sum_k w_{ik}^2 \tag{27}$$

where w_{ik} is the weight between layer i and layer k. An advantage that this saliency metric has over the one above, is that its calculation does not become significantly more difficult as the size of the perceptron grows.

Both forms of saliency are calculated in the multilayer perceptron FORTRAN program. (See Appendix B) Ruck's saliency is calculated in a subroutine called SALIENCY and Tarr's saliency metric is available by simply squaring the weights already available from the ANN subroutine. These methods of feature selection were compared and based on the results, an attempt was made to reduce the number of features used for classification.

Currently, these saliency measures are calculated and features are included subjectively based on the rank order according to the saliency measures. In order to establish a more formal procedure for determining which features are significant, a noise variable was included as a feature input along with the original inputs to represent an absolutely insignificant piece of information. Because all continuous features were normalized between zero and one, the noise was uniform (0,1). The procedure for determining significant feature inputs when a noise feature is present is outlined below.

1. Introduce a noise feature to the original set of feature vectors.
2. Determine the best architecture for the multilayer perceptron by the methods already discussed.
3. Train the network (using the optimal architecture).

4. Compute the saliency of all features (using either saliency measure).
5. Repeat steps 3 and 4 (with weights being initialized at the beginning of each training cycle).
6. Characterize the distribution of the saliency of noise based on the sample saliency values.
7. Find the pth percentile for the resulting distribution.
8. Choose only those features whose saliency is greater than this pth percentile critical value.
9. Retrain the network with the salient features.

3.4.2 Discriminant Analysis Features Stepwise Selection Procedures. In addition to the stepwise logistic regression method for the selection of features discussed above, SAS provides the STEPDISC procedure. The STEPDISC procedure uses forward, backward, or stepwise selection to produce a good discrimination model. Variables are chosen to enter or leave the model according to the following criteria:

- "the significance level of an F test from an analysis of covariance, where the variables already chosen act as covariates and the variable under consideration is the dependent variable
- the squared partial correlation for predicting the variable under consideration from the CLASS variable, controlling for the effects of the variables already selected for the model (25:1494)."

It is important that in the selection of variables for entry into the model, only one variable is considered at a time. Therefore, some combinations of variables are not evaluated and some important variables could be excluded by this procedure. A sample stepwise discriminant selection program is shown at Appendix F.

The results of the STEPDISC procedure used in this analysis were compared to the selection of input features resulting from the saliency calculations. The overall

purpose of examining various feature selection methods was to identify those features that have the greatest discriminatory capability and possibly determine some smaller set of inputs with which to train the classifiers.

Discriminant Loadings. Discriminant loadings were calculated for each of the features under study. Once again, the purpose of calculating these loadings was to order the features in terms of their contribution to classification. Discriminant loadings have the advantage over traditional approaches in that they are not as greatly affected by intercorrelations between the predictors(5).

3.5 Comparison of Methods

The overall procedure for comparing the methods of classification was to compare their estimated error rate on the validation set. Random train, test and validation sets were constructed ten times. Confusion matrices were constructed for each discrimination method and compared.

A secondary comparison was made of the feature selection methods. An attempt was made to determine the features of all methods determined as significantly important to the classification process.

In addition, a comparison was made of the approximate time required to input data, train and classify using the various classification methods. All methods developed during this analysis utilized the same computer system—SUN SPARC station 2. As stated previously, the multivariate discriminant analysis classifiers were constructed using SAS Version 6.0, and the multilayer perceptron was constructed using a FORTRAN 77 program. Since the software used for each method differed, comparisons made on the training/validation time were only approximate.

3.6 Application of Classifiers

Once the classifier with the minimum error rate for the validation set was identified, data sets were combined and the classifier was trained on the resulting

representation of the population. It was recommended that the classifier resulting from entire data set be used by each sponsor to conduct their required analysis.

IV. Results and Conclusions – Application 1: Classifying Pilot's Retention Decisions

4.1 Data Collection and Orientation

Many, many factors could account for the decision by an Air Force pilot to leave military service. In order to apply discriminant analysis techniques and a multilayer perceptron technique, it was necessary to reduce the large number of possible independent variables to something manageable. AF/DPXA has a long history of attempting to quantify the intangible characteristics of military personnel including the decision to remain in the military. As the sponsors of this research effort, AF/DPXA was in the most appropriate position to determine those characteristics of individual pilots that could possibly contribute to retention decisions. As a first step, fifty possible contributing factors were submitted to the analysts at AF/DPXA. Of these fifty, several were deemed unimportant.

The second step was to consider the availability of data. The Air Force possesses several databases containing personnel information on military personnel. These databases are maintained at several levels of the personnel organization. The database available for this research effort was maintained at Headquarters AF. Even though approximately forty factors were judged appropriate, only twenty-three were available to the analysts at AF/DPXA. Therefore, for the purposes of this analysis, only these twenty-three attributes will be analyzed. It is important to note, however, that now that the method of examining individual data elements is established, additional data elements could easily be added.

4.1.1 Data Sets A feature vector is a listing of data elements available at a specific point in time for an individual pilot eligible to separate. A data set is the combination of all feature vectors (pilots) for a specific year. Two data sets were used for this analysis— fiscal year (FY) 88 and FY 89. Each year's data set included

a "snapshot" of all pilots eligible to separate that year. The data includes the actual career decision that the pilot made by the end of the year under consideration. Each feature vector contained attributes of the individuals represented as both ordinal and categorical data.

4.1.2 Data Orientation The FORTRAN program that translates the personnel data elements for each individual pilot into continuous and binary data elements was central to the development of a classification methodology. The program steps through each data element for each individual checking for possible categorical variables and translating these variables into the appropriate binary/continuous format. Because both discriminant analysis techniques and multilayer perceptrons require numerical inputs, it was necessary to translate each of the categorical variables to an appropratie number of binary variables. For example, the data element "PME" (Professional Military Education) which was originally a single alphanumeric data element was Specifically, an individual's highest level of completed PME fits into one of four categories. This variable was translated as follows:

- Senior Service School $\Rightarrow var_1 = 1, var_2 = -1$
- Intermediate Service School $\Rightarrow var_1 = -1, var_2 = 1$
- Squadron Officer School $\Rightarrow var_1 = -1, var_2 = -1$
- Other $\Rightarrow var_1 = 1, var_2 = 1$

Figure 5 graphically depicts the organization of the data sets.

In addition to translating the variables obtained from AF/DPXA, this program adds a data element which will be referred to as "noise." This data element is simply a uniform random variable between 0 and 1. The noise variable was added to determine the effect of random noise during the training and evaluation of the classification techniques. The use of this data element for feature selection will be discussed further in future sections.

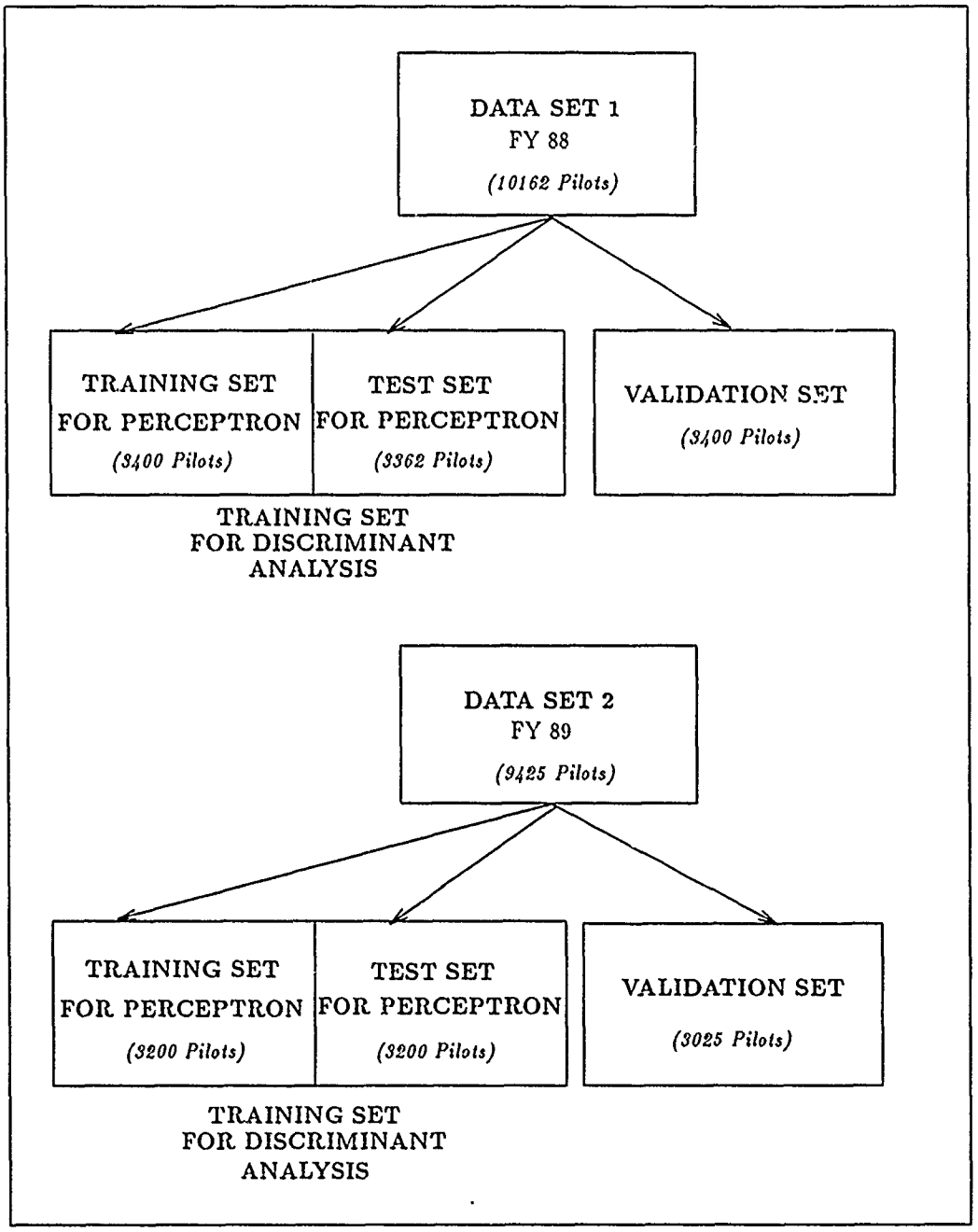


Figure 10. Data Orientation

In the development of the FORTRAN translation program the opinions of the analyst play a part. In many cases, the level of detail of the personnel data for each individual was excessive. In other cases, the available data did not capture a certain aspect of an individual. It was necessary, therefore, to decide which values of the data elements could be important to classifying the retention decision and which were meaningless. For example, the data element academic specialty (ACADSPEC) was available from AF/DPXA in a four character format which represented the detailed subject area of the academic degree that the individual received. There were over 2800 possible entries for this data element. It was determined that only the general subject area would be important to identifying those individuals who would leave military service. Therefore, for this analysis only the first two characters of the ACADSPEC data element were used. Each decision such as the one above is reflected in the FORTRAN translation program. The FORTRAN translation program is shown at Appendix C.

4.1.3 Data Elements The following is a description of the data elements used for this research effort.

1. **Total Active Federal Military Service Date (TAFMSD)** - This data element represents the date that the individual entered federal military service. This data element serves as an indicator of the years of service (YOS) an individual has completed. (Continuous)
2. **Active Duty Service Commitment Date (ADSCDA)** - This data element represents the date that the individual's last service commitment expired. This element represents the length of time the individual has been eligible to separate. (Continuous)
3. **Marital Status (MARSTAT)** - This data element gives the marital status of the individual (M=married, D=divorced, S=single). (Two binary variables)

4. **Number of Dependents (DEPN)** - This data element represents the number of dependents of each individual. Note that all children residing with the military member and the member's non-military spouse are considered dependents. (Continuous)

5. **Rated Management Code (RDTM)** - This data element shows the major category of weapon system that the individual flies. (Three binary variables)

   - Fighter = A
   - Trainer = D
   - Bomber = E
   - Tanker = F
   - Strategic Airlift = G
   - Tactical Airlift = H
   - Helicopter = J
   - Mission Support = K/L

6. **Rated Position Identifier (RPI)** - This data element describes the command level of the individual's position as well as whether the individual is in a flying or non-flying position. (Three binary variables)

7. **Grade (GRADEA)** - This data element shows the grade of the individual as two digits—"01 through 06"—for Second Lieutenant through Colonel. (Continuous)

8. **Retirement Program (RETPROG)** - This data element categorizes individuals into the different retirement programs based on the year that they entered service. (Two binary variables)

   - If the pilot entered service before August 1980, then he is in Retirement Program 1 and must serve 20 years for full retirement.

- If the pilot entered service between August 1980 and July 1986, then he is in Retirement Program 2 and must serve 25 years for full retirement.
- If the pilot entered service after July 1986, then he is in Retirement Program 3 and must serve 30 years for full retirement.

9. **Professional Military Education (PME)** – This data elements depicts the highest level of professional military education that the individual has accomplished. Although the original data element contained codes for all PME courses including those used for the enlisted grades and for the other services, the data elements were translated to the following categories:

   - Senior Service School (SSS)
   - Intermediate Service School (ISS)
   - Squadron Officer School (SOS)
   - Other

   (Two binary variables)

10. **Desired Professional Military Education (DPME)** – This data element combines the information from the PME data element and the year of service that an individual is expected to complete a specific level of PME. (One binary variable)

11. **Date of Birth (DOB)** (Continuous)

12. **Consolidated Base Personnel Office Code (PASCBPO)** – This data element specifies the personnel office where the individual's personnel actions are processed and is considered the most accurate indicator of where an individual is assigned. For the purposes of this analysis, the over 350 possible entries for this data element were reduced to one of six regions of the United States and one of four regions overseas. (Four binary variables)

    - **Region 1 – North Eastern U.S.**

- Connecticut
- Delaware
- Maine
- Maryland
- Massechusetts
- New Hampshire
- New Jersy
- New York
- Pennsylvania
- Rhode Island
- Vermont
- Virginia
- West Virginia

- **Region 2 – Southern U.S.**
  - Alabama
  - Florida
  - Georgia
  - Louisiana
  - Mississippi
  - North Carolina
  - South Carolina
  - Tennesse

- **Region 3 – Northern U.S.**
  - Minnesota
  - Montana
  - North Dakota

- South Dakota

- **Region 4 – South Western U.S.**
  - Arizona
  - Arkansas
  - New Mexico
  - Oklahoma
  - Texas
- **Region 5 – Midwestern U.S.**
  - Illinois
  - Indiana
  - Iowa
  - Kansas
  - Kentucky
  - Michigan
  - Nebraska
  - Ohio
  - Wisconsin
- **Region 6 – North Western U.S.**
  - Idaho
  - Oregon
  - Washington
- **Region 7 – Western U.S.**
  - California
  - Colorada
  - Nevada

- Utah
- Wyoming

- **Region 8 – European Area**
- **Region 9 – Pacific Area**
- **Region 10 – Central American Area**
- **Region 11 – Central Command Area**
- **Region 12 – Arctic Area**
- **Region 13 – Hawaii**
- **Region 14 – Alaska**

13. **Academic Level (ACADLVL)** - This data elements lists the level of the highest academic degree that an individual has obtained. (Three binary variables)

    - Bachelors Degree = N
    - Bachelors Degree plus Graduate Credits = O
    - Masters Degree = P
    - Masters Degree plus Post Graduate = Q
    - Doctoral Degree = R

14. **Academic Specialty (ACADSPEC)** - This data element describes the specialty area that the pilot received his most recently completed degree in (for example, electrical engineering). The possible values of this data element are too numerous to list, however, they include all major academic specialties. (Seven binary variables)

15. **Air Force Specialty Code – Duty (DAFSC)** - Air Force Regulation (AFR) 36-1 lists the possible DAFSC's for all Air Force Officers. The Air Force Specialty Code (AFSC) is a code that represents the basic grouping of positions

requiring similar skills and qualifications. To identify special skills, abilities, or equipment, a specialty may be subdivided into shredouts, identified by a letter suffix to the AFSC. Prefixes identify significant skills and abilities not restricted to a single career field. In the case of rated individuals, the *duty* AFSC describes either the weapon system that the individual is trained to fly *or* the position that the individual currently holds. Listed below are some of the more prevelant suffixes, prefixes and AFSCs contained in the data sets.

- Prefixes (Three binary variables)
  - Commander = A
  - Weapons and Tactics Instructor = S
  - Aircraft Systems Flight Evaluation = F
  - Instructor Pilot = K
  - Standardization/Flight Examiner = M
  - Squadron Operations Officer = N
  - Safety = X
- AFSCs (Four binary variables)
  - Pilot, Helicopter = 1025
  - Pilot, Search and Rescue = 1035
  - Pilot, Transport = 1045
  - Pilot, Tactical Airlift = 1055
  - Pilot, Tanker = 1065
  - Pilot, Fighter = 1115
  - Pilot, Forward Air Controller (FAC) = 1145
  - Pilot, Mission Support = 1165
  - Pilot, Strategic Bomber = 1235
  - Pilot, Special Operations = 1315

- Pilot, ElectronicWarfare/Airborne Command and Control/Special Reconnaissance = 1325
- Pilot, Strategic Reconnaissance = 1335
- Pilot, Flight Training Instructor = 1355
- Pilot, Special Operations, Helicopter = 1365
- Air Operations, Staff Director, Pilot = 1406
- Air Operations Officer, Pilot, Trainer = 1415
- Air Operations Officer, Pilot, Transport/Airlift = 1425
- Air Operations Officer, Pilot, Strategic Bomber/Tanker/Reconnaissance = 1435
- Air Operations Officer, Pilot, Tactical Air Control System = 1445
- Air Operations Officer, Pilot, Fighter = 1455
- Air Operations Officer, Pilot, Special Operations = 1465
- Air Operations Officer, Pilot, Electronic Warfare/Airborne Command and Control/Special Reconnaissance = 1475
- Air Operations Officer, Pilot, Helicopter/Search and Rescue = 1485
- Air Operations Officer, Pilot, Other = 1495

16. **Air Force Specialty Code - Primary (PAFSC)** - This data element is similar to the data element Air Force Specialty Code - Duty above, but pertains to the primary career specialty that the officer is trained to perform. Therefore, while an individual's *duty* AFSC may change often as he moves from job to job, his *primary* AFSC should rarely change. (Prefix = Three binary variables, Digit = Four binary variables)

17. **Prior Service Information (PRIORSV)** - This data element gives information on the individual's prior service in an enlisted grade. (Two binary variables)

    - No Prior Service = Category 0

- Length of Prior Service between 1 and 4 years = Category 1
- Length of Prior Service between 4 and 8 years = Category 2
- Length of Prior Service greater than 8 years = Category 3

18. **Source of Commissioning (SOC)** - This data element denotes the professional military training that the pilot received prior to commissioning. In addition, the data element reflects whether the individual was named a Distinguished Graduate (DG). (Three binary variables)

    - Reserve Officer Training Corps (ROTC)
    - ROTC DG
    - Air Force Academy (AFA)
    - AFA DG
    - Officer Training School (OTS)
    - OTS DG
    - Other
    - Other DG

19 **Race (RACE)** - This data element identifies the individual pilot's race. (Three binary variables)

    - Caucasian
    - Yellow
    - Black
    - Red
    - Other
    - Unknown

20. **Component (COMP)** - This data element identifies whether the officer is regular or reserve component. (Two binary variables)

21. **Sex (SEX)** - This data element identifies the individual pilot's sex. (One binary variable)

22. **Number of Flying Hours (FLYMONTH)** - This data element is used as an indicator of the average amount of flying that the individual pilot is performing in his current duty assignment. The value entered for this data element is the number of flying hours that the pilot flew for the month that the data was accessed. (Continuous)

23. **Separation Decision (RETAIN)** - Finally, this data element identifies whether the pilot remained in the military at the end of the year for the current data set. The value of theis data element is "1" if the individual remained in the Air Force and "0" if he separated. The RETAIN data element is the information that the classifiers are attempting to predict based on the values of all other data elements—it is the dependent variable.

After the translation of these data elements was completed, each individual's feature vector (vector of attributes) consisted of 60 binary variables and 5 continuous variables.

4.2 Simple Statistics for Input Features

Before the development of any multivariate analysis tool, an examination of the simple statistics is appropriate. Tables 1 and 2 list the input features and their means and standard deviations. Continuous variables are highlighted with '*'. The means of the binary variables serve as indicators of whether the majority of the pilots had that variable coded as a "1" or "-1." (If the mean is greater than 0 then the majority had that variable coded as "1.") "POS" in this table indicates the position of the variable in terms of the binary word that makes up the entire piece of data.

For example, since PME is composed of two binary variables, PME (1st POS) and PME (2nd POS) are listed in Tables 1 and 2. The feature VAR 14 (RETRPROG 2nd POS) is the same for all individuals in both data sets, therefore, this variable was dropped from further analysis.

4.3 Development of Classification Methodology

The following section describes the process used to generate discriminators to classify Air Force pilots into one of two groups—"stay" or "leave".

4.3.1 Application of Multilayer Perceptron Techniques Initially, the entire set of 65 input features was used to develop a multilayer perceptron for classification. The methodology for constructing an optimal multilayer perceptron architecture is shown in Figure 8. The initial "standardized" parameters for the network were set to the following values

- **Number of Middle Nodes: 25**
- **Learning Rate: 0.30**
- **Momentum Rate: 0.70**
- **Epochs: 2000**
- **Data Set: FY 88 Pilot Data (All Features)**

For the 2000 epochs that the multilayer perceptron was presented data, no decrease in error is observed, and therefore, it appears that no training took place. The error plots in Figures 11 and 12 illustrate this lack of training. As stated earlier, it is expected that two types of error should decrease as the number of epochs increases—output error and classification error.

The results obtained with this "standard" structure illustrate the characteristics that occur when a multilayer perceptron fails to learn, or decrease error. The classification error varies around .5 signifying that the multilayer perceptron was

Table 1. FY 88 and FY 89 Pilot Data (Features 1-36) – Simple Statistics

FEATURE NUMBER	FEATURE NAME	MEAN	STANDARD DEVIATION
VAR1*	TAFMCSD	7141.83	599.16
VAR2*	ADSCDA	8751.13	184.77
VAR3	MARSTAT (1st POS)	-0.8966	0.4429
VAR4	MARSTAT (2nd POS)	-0.9264	0.3766
VAR5*	DEPN	2.3366	1.4376
VAR6	RDTM (1st POS)	0.3327	0.9431
VAR7	RDTM (2nd POS)	0.3220	0.9468
VAR8	RDTM (3rd POS)	0.2755	0.9613
VAR9	RPI (1st POS)	0.0462	0.9990
VAR10	RPI (2nd POS)	0.5050	0.8632
VAR11	RPI (3rd POS)	0.4937	0.8697
VAR12	GRADE	4.2910	1.0730
VAR13	RETPROG (1st POS)	-0.8161	0.5779
VAR14	RETPROG (2nd POS)	-1.000000	0
VAR15	PME (1st POS)	-0.1862	0.9825
VAR16	PME (2nd POS)	0.0898	0.9960
VAR17	DPME	-0.4510	0.8926
VAR18	MAJCOM (1st POS)	-0.2401	0.9708
VAR19	MAJCOM (2nd POS)	0.1356	0.9908
VAR20	MAJCOM (3rd POS)	0.0681	0.9977
VAR21	MAJCOM (4th POS)	-0.2746	0.9616
VAR22*	DOB	4851.42	595.70
VAR23	PASCBPO (1st POS)	0.0852	0.9964
VAR24	PASCBPO (2nd POS)	0.3051	0.9523
VAR25	PASCBPO (3rd POS)	0.5450	0.8385
VAR26	PASCBPO (4th POS)	-0.7255	0.6883
VAR27	ACADLVL (1st POS)	0.8596	0.5109
VAR28	ACADLVL (2nd POS)	-0.2591	0.9659
VAR29	ACADLVL (3rd POS)	0.9661	0.2582
VAR30	ACADSPEC (1st POS)	-0.1832	0.9831
VAR31	ACADSPEC (2nd POS)	0.3935	0.9194
VAR32	ACADSPEC (3rd POS)	-0.4268	0.9044
VAR33	ACADSPEC (4th POS)	0.5526	0.8335
VAR34	ACADSPEC (5th POS)	0.3613	0.9325
VAR35	ACADSPEC (6th POS)	0.7885	0.6151
VAR36	ACADSPEC (7th POS)	0.4963	0.8682

'*' indicates continuous variable

Table 2. FY 88 Pilot Data (Features 37-65) – Simple Statistics

FEATURE NUMBER	FEATURE NAME	MEAN	STANDARD DEVIATION
VAR37	DAFSCPRE (1st POS)	-0.6908	0.7231
VAR38	DAFSCPRE (2nd POS)	-0.6911	0.7228
VAR39	DAFSCPRE (3rd POS)	-0.8344	0.5512
VAR40	DAFSCDIGIT (1st POS)	-0.6968	0.7173
VAR41	DAFSCDIGIT (2nd POS)	-0.6780	0.7350
VAR42	DAFSCDIGIT (3rd POS)	-0.6160	0.7878
VAR43	DAFSCDIGIT (4th POS)	-0.4324	0.9017
VAR44	DAFSCDIGIT (5th POS)	-0.7135	0.7007
VAR45	PAFSCPRE (1st POS)	-0.6843	0.7292
VAR46	PAFSCPRE (2nd POS)	-0.6598	0.7515
VAR47	PAFSCPRE (3rd POS)	-0.8365	0.5480
VAR48	PAFSCDIGIT (1st POS)	-0.7145	0.6996
VAR49	PAFSCDIGIT (2nd POS)	-0.6971	0.7170
VAR50	PAFSCDIGIT (3rd POS)	-0.6428	0.7661
VAR51	PAFSCDIGIT (4th POS)	-0.4732	0.8810
VAR52	PAFSCDIGIT (5th POS)	-0.7314	0.6819
VAR53	PRIORSV (1st POS)	-0.9065	0.4223
VAR54	PRIORSV (2nd POS)	0.9585	0.2852
VAR55	SOC (1st POS)	0.0545	0.9986
VAR56	SOC (2nd POS)	-0.2420	0.9703
VAR57	SOC (3rd POS)	-0.4188	0.9081
VAR58	RACE (1st POS)	-0.9675	0.2531
VAR59	RACE (2nd POS)	-0.9877	0.2001
VAR60	RACE (3rd POS)	-0.9798	0.2001
VAR61	COMP (1st POS)	-0.9600	0.2800
VAR62	COMP (2nd POS)	-0.9632	0.2688
VAR63	SEX	-0.9840	0.1780
VAR64*	FLYMONTH	144.47	59.51
VAR65*	NOISE	0.4960	0.2883

'*' indicates continuous variable

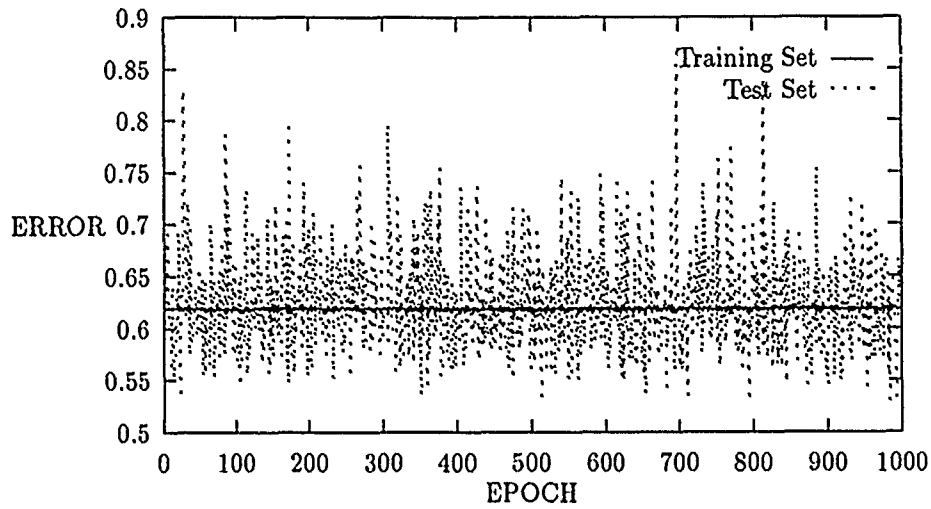


Figure 11. FY 88 Pilot Data (All Features) - Output Error for Structure 1

performing no better than guessing the group membership. All attempts to follow the procedure outlined in Chapter 3 for determining the optimal parameters for the multilayer perceptron failed, since a trained net is required to evaluate changes in the parameters. Instead, the analyst was left to try different combinations of parameters hoping that some training takes place.

The confusion matrix for the FY 88 training set at the time that training ceased is shown in Table 3. At this time in the training cycle, the rate for predicting those pilots whose true classification is Group 1 (stay) was very good, but very bad for those whose true classification was Group 2 (leave). In fact, the network classified all but 37 individuals as staying.

Since there were such a large number of inputs (65 features) to the multilayer perceptron, it was suggested that the number of middle nodes be increased. In practice, multilayer perceptrons are often diamond shaped with the number of middle nodes being greater than the number of inputs and the number of output nodes being smaller than the number of middle nodes (21). Also, since there were 65 inputs to the perceptron, there was the possibility that the learning rate and

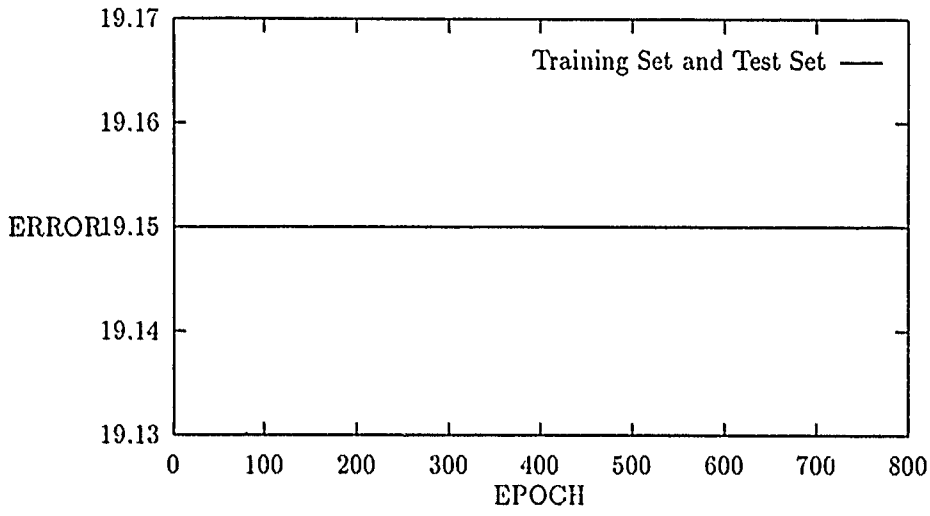


Figure 12. FY 88 Pilot Data (All Features) - Classification Error for Structure 1

Table 3. FY 88 Pilot Data (All Variables) - Confusion Matrix for Structure 1

TRAINING SET			
	Classified Stay	Classified Leave	Total
True Stay	100% 2749	0% 0	100% 2749
True Leave	100% 651	0% 0	100% 651
Total	3400	0	3400
Total Training Set Error Rate: 0.1915			

momentum were set too high. A learning rate that is too high will cause the weights to vary greatly around the minimum of the error surface while never actually finding the minimum. Subsequently, these parameters were decreased and the following structure was established:

- **Number of Middle Nodes: 100**
- **Learning Rate: 0.015**
- **Momentum Rate: 0.0**
- **Epochs: 3000**
- **Data Set: FY 88 Pilot Data (All Features)**

Figures 13 and 14 show the output and classification error for the multilayer perceptron described. Once again, the output error did not decrease and the classification error varied around 50% correct. In fact, the output error for the training set varied very little suggesting that the perceptron has reached some minimum error and cannot decrease the error. The classification error suggests that the multilayer perceptron has trained to weights which classify *all* pilots as "stay" and that this is the minimum error solution.

The confusion matrix for the multilayer perceptron for the training set at the time that training ceased is shown in Table 4. Again, the network was classifying nearly all feature vectors as Group 1.

At this stage, the validation set error rates were calculated as if the multilayer perceptron was fully trained. The resulting error rates were 20.17% incorrect and 25.26% incorrect. These rates, taken by themselves, suggest that a relatively accurate classifier was developed. These rates are deceiving, however, due to the lack of discrimination of the pilots who left the Air Force.

In addition to the two structures described above, training of the multilayer perceptron was attempted using several different learning rates, momentum rates, number of middle nodes, and epochs. Table 5 details the ranges of the parameters

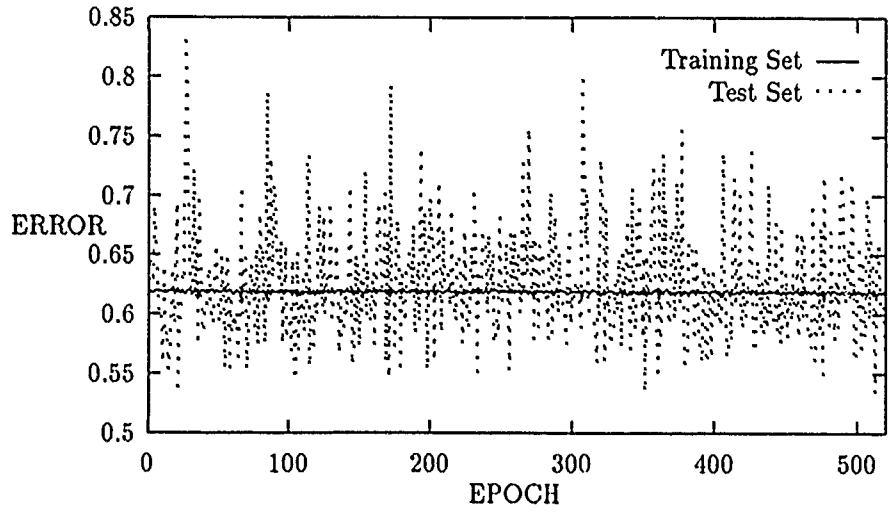


Figure 13. FY 88 Pilot Data (All Features) - Output Error for Structure 2

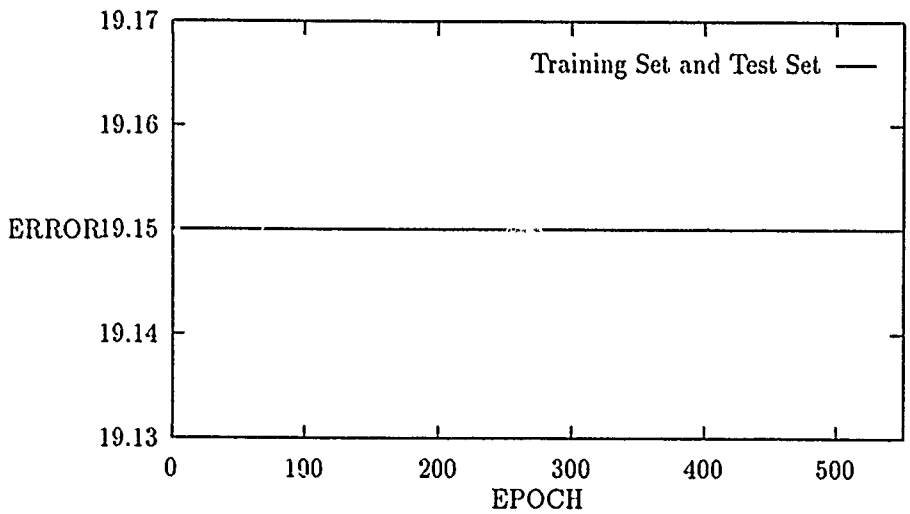


Figure 14. FY 88 Pilot Data (All Features) - Classification Error for Structure 2

Table 4. FY 88 Pilot Data (All Variables) - Confusion Matrix for Structure 2

TRAINING SET			
	Classified Stay	Classified Leave	Total
True Stay	100% 2749	0.0% 0	100% 2749
True Leave	100% 651	0.0% 0	100% 651
Total	3400	0	3400
Total Training Set Error Rate: 0.1915			

Table 5. Multilayer Perceptron Parameters for Pilot Data Sets

Number of Middle Nodes	5–100
Learning Rate	0.01–0.70
Momentum Rate	0.00–0.70
Epochs	1–10,000

that were explored. It should be emphasized that the time required to attempt to train the multilayer perceptron over this range of parameters was excessive. For the hardware being used, the run time for each attempt was well over 72 hours. In the runs with an especially large numbers of epochs, the program continued to run and the reliability of the hardware was the reason for stopping.

By the end of the analysis effort, an attempt was made to train approximately 15 different multilayer perceptron structures. The error plots in Figures 11 through 14 are representative of *all* error plots for *all* parameters.

The confusion matrices for these attempts suggest that during training, the weights converged to values that classified all feature vectors as Group 1 (stay). This could be due to the fact a larger percentage of the data presented to the

multilayer perceptron was Group 1 data. In fact, for the FY 88 data set only 19.2 percent of the data represents individuals that left the Air Force—Group 2. A neural network considers the data set used to train its weights as representative of the entire population. Consequently, networks assume that the prior probabilities of group membership are proportional to the membership in the training set. In this application, it is possible that the probability of membership in Group 2 was not large enough to cause the network to adjust its weights to account for Group 2.

Another explanation for the lack of training and the large errors is that the two groups are not separable. A basic assumption of all discriminators used in this research effort was that the groups to be classified were separable. SAS provides procedures to determine if the group centroids are significantly different—specifically, the Hotelling-Lawley Trace test. The results of this test showed that the null hypothesis that the means were equal was rejected (p-value = .0001). According to this test, the means are significantly different. However, the test is based on an F-statistic with 6695 denominator degrees of freedom. With this many degrees of freedom, it would be difficult to do anything except reject the null hypothesis. The hypothesis that the groups are not separable was still unproven.

To examine this non-separability issue further, each of the 65 features was standardized between 1 and 0 and the location of the standardized group centroids was determined using SAS. The squared distance between the centroids was found to be 1.51135. Since all of the features were standardized between 0 and 1, the greatest that the squared distance between the centroids could be is 65. The relatively small distance between the standardized Group 1 and Group 2 centroids strengthens the argument that the groups are not significantly different *in the current feature space.*

Conclusion: The multilayer perceptron with 65 features as inputs could not be trained to discriminate between pilots that will leave and those that will remain in the Air Force.

4.3.2 Application of Logistic Regression The LOGISTIC function in SAS provides a user friendly environment for constructing a logistic regression model for the two group problem. The only decision required by the user is to specify the significance levels for the entry and removal of variables from the model. SAS provides the user with information on the feature inputs themselves, as well as parameter estimates, standardized parameter estiamtes, Chi-square scores, a confusion matrix and an estimate of the error rate.

Tables 6 and 7 show the confusion matrices for the logistic regression technique on FY 88 and FY 89 training sets with all 65 variables. The total rate correct for all feature vectors is 81.0% for the FY 88 data set and 76.4% for the FY 89 data set. These rates appeared favorable, however, further investigation of the confusion matrices was necessary.

As in the case of the multilayer perceptron technique, the percentage of vectors classified as 'stay' that were truly 'stay' was high while the percentage classified as 'leave' that were truly 'leave' is low. One might optimistically look at the confusion matrix and notice that there were 203 individuals correctly identified as leaving for the FY 88 data. In the context of this application, however, this identification rate is unacceptable. It would mean underestimating those pilots that will leave the Air Force and could be detrimental to analyses of force projections.

The reported non-error rates in the confusion matrices for the training sets are optimistic since the same set was used to construct the discriminator and to eveluate its performance. Because poor performance was observed for the training set, no decrease in error was expected for the validation set. For this reason, the validation set was not examined.

Conclusion: The logistic regression methodology using 65 features produced a classifier with an error rate of 19.0% for the FY88 data set and 23.6% for the FY 89 data set. The classifier performed poorly when classifying those individuals that left the Air Force.

Table 6. Pilot Data FY 88 – Logistic Regression Confusion Matrix

TRAINING SET			
	Classified Stay	Classified Leave	Total
True Stay	96.5% 5270	3.5% 193	100% 5463
True Leave	84.3% 1092	15.7% 203	100% 1295
Total	6362	396	6758
Total Training Set Error Rate: 0.1901			

Table 7. Pilot Data FY 89 – Logistic Regression Confusion Matrix

TRAINING SET			
	Classified Stay	Classified Leave	Total
True Stay	92.5% 4400	7.5% 357	100% 4757
True Leave	70.4% 1152	29.6% 484	100% 1636
Total	5552	841	6393
Total Training Set Error Rate: 0.2360			

4.3.3 Application of K-Nearest-Neighbor. The nonparametric option available in DISCRIM was used to test the k-nearest-neighbor algorithm for $k = 1$ through $k = 9$. The nonparametric form of DISCRIM provides the following information

- Class Level Information
  - Frequency of each Group's Membership
  - Prior Probability of Group Membership
- Resubstitution Confusion Matrix. Resubstitution presents the same data to both develop the classifier and evaluate the discriminator. The resulting error rate is optimistically biased (25:685).
- Cross-Validation Confusion Matrix. Crossvalidation treats $n-1$ out of n training observations as a training set. It determines the discriminator based on these $n-1$ observations and then applies the discriminator to the one observation left out. This is done for n training observations. The misclassification rate for each group is the proportion of sample observations in that group that were incorrectly classified. This method yields a nearly unbiased error estimate, however, the estimate has a high variance (25:685).
- Validation Set Confusion Matrix. Validation presents the classifier with a data set that was not used to train the classifier. This method gives the most honest estimate of error rate, but decreases information available to train the classifier.

Tables 8 and 9 show the total error rates for each value of k. Since larger values of k smooth the region defining each of the groups, it follows that the larger values of k will generalize better. For the values of k chosen, $k = 1$ perfectly classifies the data in the training set, however, the validation sets error rate is highest for $k = 1$. The nearest neighbor algorithm with $k = 1$ appeared to provide the least generalization and $k = 9$ appeared to provide the most generalization. For the FY 88 and FY 89 data sets, $k = 7$ was chosen. (The validation set error rate for $k = 7$ was very close

Table 8. Pilot Data FY 88 - Values of k for k-Nearest Neighbor Neighbor Technique

k	Resubstitution Error Rate	Crossvalidation Error Rate	Validation Error Rate
1	0.00%	26.12%	27.02%
3	13.02%	21.60%	22.23%
5	15.57%	19.99%	20.57%
7	16.37%	19.03%	19.82%
9	16.91%	18.60%	19.82%

Table 9. Pilot Data FY 89 - Values of k for k-Nearest Neighbor Neighbor Technique

k	Resubstitution Error Rate	Crossvalidation Error Rate	Validation Error Rate
1	0.00%	31.69%	31.40%
3	16.22%	27.86%	27.57%
5	19.18%	25.65%	26.31%
7	20.01%	24.28%	25.09%
9	20.96%	24.09%	24.28%

to the rate for $k = 9$, while $k = 7$ had a lower resubstitution error rate than the $k = 9$ resubstitution error rate.)

Tables 10 and 11 shows the confusion matrices for the 7-nearest-neighbor algorithm for the FY 88 and FY 89 data sets.

Once again, the k-nearest neighbor algorithm provided accurate classifications of those individuals who remained in military service, however, it classified poorly those who left. Again, several individuals were classified correctly as leaving the Air Force (82 individuals for the FY 88 validation set and 178 individuals for the FY 89 validation set). These results were not viewed as successful in light of the large number of individuals incorrectly classified.

Table 10. Pilot Data FY 88 - K-Nearest-Neighbor Confusion Matrices

RESUBSTITUTION			
	Classified Stay	Classified Leave	Total
True Stay	97.84% 5345	2.16% 118	100% 5463
True Leave	76.29% 988	23.71% 307	100% 1295
Total	6333	425	6758

CROSSVALIDATION			
	Classified Stay	Classified Leave	Total
True Stay	96.54% 6371	3.46% 387	100% 5463
True Leave	84.71% 1097	15.29% 198	100% 1295
Total	6371	387	6758

VALIDATION			
	Classified Stay	Classified Leave	Total
True Stay	96.35% 2612	3.65% 99	100% 2711
True Leave	88.03% 603	11.97% 82	100% 685
Total	3215	181	3396
Total Validation Set Error Rate: 0.2067			

Table 11. Pilot Data FY 89 – K-Nearest-Neighbor Confusion Matrices

RESUBSTITUTION			
	Classified Stay	Classified Leave	Total
True Stay	95.38% 4537	4.62% 220	100% 4757
True Leave	64.73% 1059	35.27% 577	100% 1636
Total	5596	797	6393

CROSSVALIDATION			
	Classified Stay	Classified Leave	Total
True Stay	93.08% 4428	6.92% 329	100% 4757
True Leave	74.76% 1223	25.24% 413	100% 1636
Total	5651	742	6393

VALIDATION			
	Classified Stay	Classified Leave	Total
True Stay	92.65% 2091	7.35% 166	100% 2257
True Leave	76.67% 585	23.33% 178	100% 763
Total	2676	344	3020
Total Validation Set Error Rate: 0.2487			

The individuals that were correctly identified as leaving the Air Force were examined to determine if they possessed characteristics that differed greatly from the rest of the population. The only significant difference detected was an unusually high number of unmarried individuals with no dependents. That this characteristic affects retention decisions is reasonable since individuals without families are more likely to risk possible unemployment after leaving the Air Force.

Conclusion: The k-nearest-neighbor algorithm using 65 features produced a classifier with a total error of 20.67% for the FY 88 validation set and 24.87% for the FY 89 validation set. These error rates are likely to be unacceptable considering the poor performance classifying those that left the Air Force.

4.4 Feature Selection

4.4.1 Multilayer Perceptron Feature Selection Results. The failure of the perceptron to accurately classify pilots highlights one of the major disadvantages of neural networks. All of the techniques for evaluating the significance of input features are based on a *trained network.* Therefore, saliency cannot be calculated and attempts to include high-order terms are not possible. It was decided that a preprocessing method to reduce the number of features would be completed outside the realm of neural networks. This leads to the next type of feature selection that was developed, the logistic regression model.

Conclusion: Without a trained multilayer perceptron, the number of features cannot be reduced by the saliency methods proposed for this research effort.

4.4.2 Logistic Regression Feature Selection Results. The SAS LOGISTIC procedure provided stepwise selection of features as well as chi-square scores and the order of feature entry and removal. The levels of significance used for entry and

removal of variables was 0.3. The results of the stepwise selection procedure are shown in Tables 12 and 13. It was at this stage of the analysis that a uniform (0,1) noise was added.

It is immediately obvious that the need to decompose the personnel data on each individual into binary words puts the analyst great disadvantage. According to Tables 12 and 13, only certain portions of the binary words are significant for the classification process. The analyst was faced with the decision of which variables to use and an evaluation of the meaning of teh variables chosen.

The advantage of including a variable representing noise is evident in the stepwise selection results. In both the FY 88 and FY 89 data, SAS entered the noise variable to the model as though it contributed to the classification. This suggests that all variables entered after noise contribute to the classification no more than pure noise and *should not be included.*

The SAS LOGISTIC model was constructed again, this time including only the top ten variables that were chosen in the selection process (i.e., the top ten variables in the lists in Tables 12 and 13.) Tables 14 and 15 show the confusion matrix for the resulting reduced models. Very little change from the models with 65 inputs was observed for these reduced models.

Initially, it was thought that the logistic regression stepwise procedure could be used as a pre-processor for the multilayer perceptron. To see if the multilayer perceptron could be trained with the logistic regression reduced data set, an attempt was made to train the multilayer perceptron with the 10 most significant features. Several multilayer perceptron architectures were tested and all attempts to train the network failed.

To see if this reduced data set may produce a better classifier with the k-nearest neighbor algorithm, the reduced data set was presented to the algorithm. A with the original data set, $k = 7$ produced the classifier with the least validation set

Table 12. Logistic Regression Stepwise Selection Results - FY 88

Step	Variable Entered	Variable Removed	Number In	Score Chi-Square	Variable Name
1	VAR13		1	283.8	RETPROG
2	VAR2		2	133.9	ADSCDA
3	VAR52		3	85.2374	PAFSC DIGIT (5th POS)
4	VAR28		4	70.5581	ACADLVL (2nd POS)
5	VAR16		5	62.1938	PME (2nd POS)
6	VAR27		6	38.1435	ACAD LVL (1st POS)
7	VAR5		7	23.5200	DEPN
8	VAR22		8	21.2534	DOB
9	VAR12		9	62.2929	GRADE
10	VAR17		10	22.5306	DPME
11	VAR4		11	13.1699	MARSTAT
12	VAR15		12	9.8369	PME (1st POS)
13	VAR24		13	9.7845	PASCBPO (2nd POS)
14	VAR57		14	7.8916	SOC (3rd POS)
15	VAR10		15	6.8891	RPI (2nd POS)
16	VAR32		16	4.7363	ACADSPEC (3rd POS)
17	VAR34		17	5.2052	ACADSPEC (5th POS)
18	VAR1		18	4.0425	TAFMSD
19	VAR53		19	4.4254	PRIORSV (1st POS)
20	VAR45		20	4.2777	PAFSC PRE (1st POS)
21	VAR47		21	6.3393	PAFSC PRE (3rd POS)
22	VAR55		22	3.1298	SOC (1st POS)
23	VAR11		23	2.9734	RPI (3rd POS)
24	VAR9		24	8.4733	RPI (1st POS)
25	VAR38		25	2.6946	DAFSC PRE (2nd POS)
26	VAR37		26	2.7555	DAFSC PRE (1st POS)
27	VAR25		27	1.8429	PASCBPO (3rd POS)
28	VAR40		28	1.5299	DAFSC DIGIT (1st POS)
29	VAR18		29	1.5282	MAJCOM (1st POS)
30	VAR65		30	1.5036	**** NOISE ****
31	VAR63		31	1.4217	SEX
32	VAR56		32	1.3269	SOC (2nd POS)
33		VAR57	31		SOC (3rd POS)
34	VAR54		32	1.3793	PRIORSV (2nd POS)
35	VAR46		33	1.1866	PAFSC PRE (2nd POS)
36		VAR45	32		SOC (2nd POS)

Table 13. Logistic Regression Stepwise Selection Results - FY 89

Step	Variable Entered	Variable Removed	Number In	Score Chi-Square	Variable Name
1	VAR13		1	435.4	RETPROG
2	VAR1		2	135.7	TAFMSD
3	VAR12		3	239.0	GRADE
4	VAR5		4	27.3405	DEPN
5	VAR44		5	21.4935	DAFSC DIGIT (5th POS)
6	VAR63		6	15.6870	SEX
7	VAR28		7	14.7993	ACAD LVL (2nd POS)
8	VAR25		8	13.0541	PASCBPO (3rd POS)
9	VAR37		9	10.4393	DAFSC PRE (1st POS)
10	VAR16		10	8.9730	PME (2nd POS)
11	VAR17		11	29.7456	DPME
12	VAR54		12	13.8000	PRIORSV (2nd POS)
13	VAR15		13	8.6156	PME (1sy POS)
14	VAR59		14	7.4902	RACE (2nd POS)
15	VAR29		15	7.0275	ACAD LVL (3rd POS)
16	VAR62		16	5.7425	COMP (2nd POS)
17	VAR10		17	5.8251	RPI (2nd POS)
18	VAR53		18	5.1975	PRIORSV (1st POS)
19	VAR9		19	4.8152	RPI (1st POS)
20	VAR11		20	7.7491	RPI (3rd POS)
21	VAR64		21	3.7386	FLYMONTH
22	VAR18		22	3.8304	MAJCOM (1st POS)
23	VAR6		23	3.2256	RDTM (1st POS)
24	VAR65		24	3.0706	*****NOISE *****
25	VAR2		25	2.5743	ADSCDA
26	VAR23		26	2.3577	PASCBPO (1st POS)
27	VAR56		27	2.2657	SOC (2nd POS)
28	VAR7		28	2.3411	RDTM (2nd POS)
29	VAR20		29	3.5173	MAJCOM (3rd POS)
30	VAR40		30	2.8548	DAFSC DIGIT (1st POS)
31	VAR31		31	2.0140	ACAD SPEC (2nd POS)
32	VAR55		32	1.8805	SOC (1st POS)
33	VAR50		33	1.7196	PAFSC DIGIT (3rd POS)
34	VAR43		34	3.6511	DAFSC DIGIT (4th POS)
35		VAR44	33		DAFSC DIGIT (5th POS)
36	VAR52		34	1.1990	PAFSC DIGIT (5th POS)

Table 14. Pilot Data FY 88 – Logistic Regression Confusion Matrix (Reduced Feature Set)

TRAINING SET			
	Classified Stay	Classified Leave	Total
True Stay	97.13% 5306	2.87% 157	100% 5463
True Leave	86.50% 1121	13.50% 175	100% 1296
Total	6427	332	6759
Total Training Set Error Rate: 0.1891			

Table 15. Pilot Data FY 89 – Logistic Regression Confusion Matrix (Reduced Feature Set)

TRAINING SET			
	Classified Stay	Classified Leave	Total
True Stay	92.00% 4382	8.00% 381	100% 4763
True Leave	71.47% 1170	28.53% 467	100% 1637
Total	5552	848	6400
Total Training Set Error Rate: 0.2423			

errors. The resulting 7-nearest neighbor classifier with the reduced data set produced slightly higher error rates for the FY 88 and 89 data than the original input feature set.

4.4.3 Other Discriminant Analysis Feature Selection Results. The results of the SAS STEPDISC procedure and the discriminant loading calculations looked very much like the logistic regression stepwise results. Although the selection results were not exactly the same, all methods chose the same ten most significant features. For this reason, no further analysis was conducted on these feature selection method.

Conclusion: Feature selection techniques using multilayer perceptrons were not possible since the network failed to train on the data. Feature selection using parametric methods (logistic regression and stepwise discriminant analysis) produced significant features, although the reduced feature set did not reduce the error rates.

4.5 Comparison of Methods

4.5.1 Classification Accuracy. None of the methodologies performed to levels that could be considered successful. The multilayer perceptron did not exhibit learning, however, the classification error rate of the perceptron on the FY 88 validation set was 20.17% and 25.26% for the FY 89 validation set. The logistic regression model produced overall classification error rates on the training set of between 18.9% and 24.2%. The 7-nearest-neighbor algorithm produced classification error rates for the FY 88 validation set of 20.67% and 24.87% for the FY 89 validation set. Note that the multilayer perceptron achieved error rates comparable to the k-nearest-neighbor algorithm by classifying *all* pilots as into a single group—"stay."

Although the nearest-neighbor algorithm performed similarly to the multilayer perceptron, it should be highlighted that this nearest-neighbor methodology would be the most difficult to implement by AF/DPXA. The multilayer perceptron and the

logistic regression methodologies produce weights or parameter estimates once they have been trained. These coefficients are all that is needed to classify new pieces of data that the classifiers have not seen. The k-nearest neighbor algorithm, on the other hand, requires a comparison with *all* the points in the training set to classify a new observation.

In terms of the ability to reduce the number of features in the model, the logistic regression model obviously outperformed the perceptron methods. The multilayer perceptron feature selection techniques require a trained network which, as has been shown, can be difficult to achieve. The k-nearest-neighbor algorithm has no ability to identify significant inputs.

Conclusion: The 7-nearest-neighbor classification methodology exhibited the least classification error rate, however, the error that was observed was significant.

4.5.2 Model Complexity. It was difficult to compare the complexity of the models used for this application. All three classification methodologies required large amounts of data translation and data manipulation. It seemed that the methodologies developed with the SAS package were simpler to implement and yet more difficult to interpret. The multilayer perceptron methodology was fully understood since it was within the scope of this research effort to actually develop this classifier.

If run time was used as the only measure of complexity for these classifiers, then the multilayer perceptron was extremely complex. To run the software on the SPARC station 2 required three days for the standard run (i.e., the initial perceptron architecture). Both the logistic regression and the k-nearest neighbor methodologies run on the same system required at most 5 hours. An investigation of the underlying methodology, however, suggests that the logistic regression model is the most complex to compute due to the use of the IRLS algorithm to determine the parameters. Finally, the k-nearest-neighbor classification methodology is perhaps the most

difficult of all three to employ. To classify a previously unseen feature vector using this method, it is necessary to have all of the data point available so that distance comparisons can be made.

Conclusion: The multilayer perceptron technique was too computationally complex to produce a solution in a reasonable amount of time. The methods performed with SAS were computationally more efficient.

4.6 Application of Classifiers

Since none of the three methods produced successful classifiers, it was not appropriate to present the entire data sets to a single classifier and produce results that could be used by AF/DPXA.

V. Results and Conclusions – Application 2: Classifying API Projectile Complete Incendiary Functioning

5.1 Data Collection and Orientation

As with the pilot retention rate application, many factors could account for the functioning or nonfunctioning of an API projectile. SEB chose a standard penetration mechanics' testing sequence to obtain the data from each of the firings. SEB fired 144 small millimeter and 141 larger millimeter API projectiles against the composite test panels. Three projectile striking velocities (V_s) and five obliquity angles (OBL) were chosen to provide a range of impact conditions.

5.1.1 Data Set. A feature vector for this application is a listing of the parameters of each API projectile firing. Each feature vector also contains the actual classification of the firing as complete or other. The data set for this application contains all of the feature vectors available from SEB for API projectiles impacting a specific material. Figure 15 depicts the organization of the API projectile data.

All of the features for this application were measured, continuous variables, and therefore, the problems with categorical variables encountered in the first application do not apply.

5.1.2 Data Elements.

1. **Striking Velocity (V_s)** – The projectile's velocity was measured immediately before impact and was assumed to be the striking velocity of the projectile on the panel.
2. **Obliquity Angle (OBL)** – The obliquity angle is the angle between the perpendicular to the panel surface and the projectile's flight path. For example, a shot fired straight at a panel has a zero degree angle of obliquity. In this

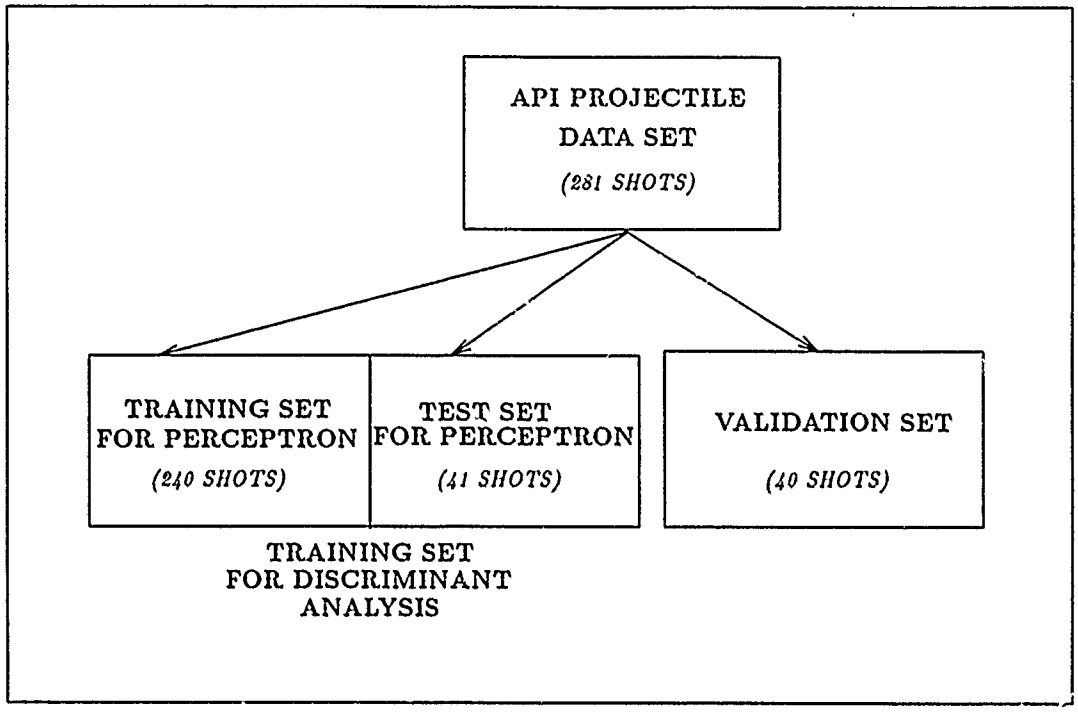


Figure 15. Data Orientation for API Projectile Data

analysis, OBL is converted to the secant of the angle, a commonly accepted practice in penetration mechanics analyses. The new variable's name is SECT.

3. **Striking Mass** (M_s) – The striking mass is assumed to be the mass of the projectile before firing and is measured in grains.

4. **Panel Ply Thickness** ($TKIN$) – Test panels were approximately 8" × 8" in size and varied in thickness according to ply size. The three plys, 32, 48, and 64 are approximately .16, .24, and .32 inches respectively.

5. **Incendiary Functioning** (IF) – The incendiary mixture flash of an API is known as incendiary functioning. Pictures provided by flash photography allowed engineers to classify a projectile's type of incendiary functioning. For this analysis, projectile firings will be classified as complete or other.

Table 16. API Projectile Data – Simple Statistics

FEATURE NUMBER	FEATURE NAME	UNITS	MEAN	STANDARD DEVIATION
VAR 1	PLY	Ply	47.8672	13.3859
VAR 2	VS	Grains	2127.2739	391.8957
VAR 3	MS	Feet/Sec	870.3141	121.4655
VAR 4	SECT	Radians	1.7687	0.7186

5.2 Simple Statistics for Input Features

Table 16 shows the simple statistics for the API projectile parameters.

5.3 Development of Classification Methodology

The following section describes the process used to generate discriminators to classify API projectiles into one of two groups—"complete" or "other."

5.3.1 Application of Multilayer Perceptron Techniques Multilayer Perceptron Training Results. In this second application, the network was initially trained with what are considered standard parameters for this size application. The parameters for this "standard" perceptron are listed below

- **Number of Middle Nodes: 10**
- **Learning Rate: 0.30**
- **Momentum Rate: 0.70**
- **Epochs: 3000**
- **Data Set: API Projectile Data**

The procedure outlined in Chapter 3 Figure 8 for determining the optimal parameters was followed to arrive at the optimal multilayer perceptron structure for the problem of classifying API projectiles. It was necessary to train approximately 12 networks to arrive at the following optimal configuration.

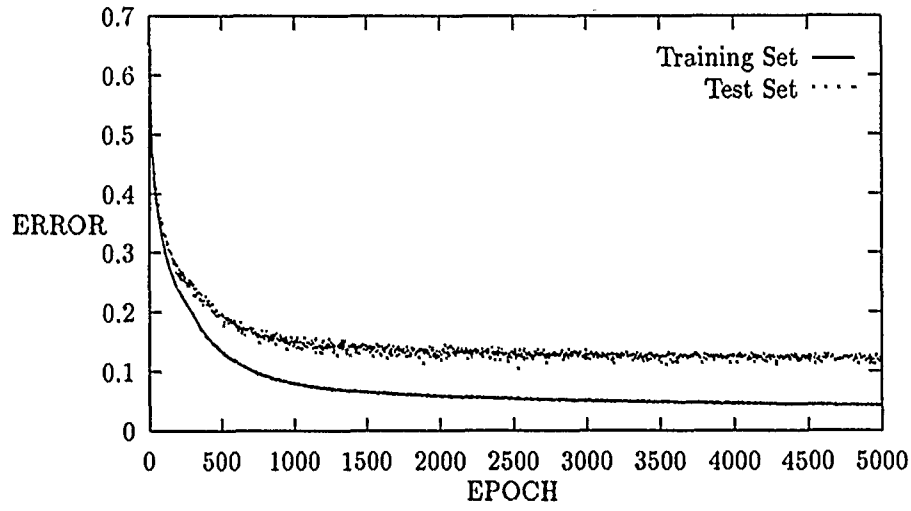


Figure 16. API Projectile Data – Output Error for Optimal Structure

- **Number of Middle Nodes: 20**
- **Learning Rate: 0.20**
- **Momentum Rate: 0.00**
- **Epochs: 5000**
- **Data Set: API Projectile Data**

Due to the random assignment of vectors to the three data sets, the initialization of the weights and the the random presentations of training vectors, error rates for a trained optimal network were slightly different for each training cycle. To accurately estimate the error rates for the training set, test set, and validation set, the optimal network was trained 10 times with 10 different divisions of the data set and an average calculated. The confusion matrices for the three data sets are shown in Table 17. Figures 16 and 17 show the classification error and output error for this optimal parameter setting. For this network, the output and classification error began to decrease almost immediately signifying that training was taking place.

Conclusion: The multilayer perceptron classified the data in the validation set with an error rate of 4.5%.

Table 17. API Projectile Data - Confusion Matrices for Optimal Multilayer Perceptron Structure (Average Over 10 Runs)

TRAINING SET			
	Classified Complete	Classified Other	Total
True Complete	96.67% 42.2	3.33% 0.1	100% 42.3
True Other	0.75% 1.2	99.25% 156.5	100% 157.7
Total	47.3	152.7	200

TEST SET			
	Classified Complete	Classified Other	Total
True Complete	86.12% 8.2	13.88% 1.3	100% 9.5
True Other	3.81% 30.3	96.19% 1.2	100% 31.5
Total	38.5	2.5	41

VALIDATION SET			
	Classified Complete	Classified Other	Total
True Complete	87.95% 7.3	12.05% 1	100% 8.3
True Other	2.52% 0.8	97.48% 30.9	100% 31.7
Total	8.1	31.9	40.0
Total Validation Set Error Rate: .0450			

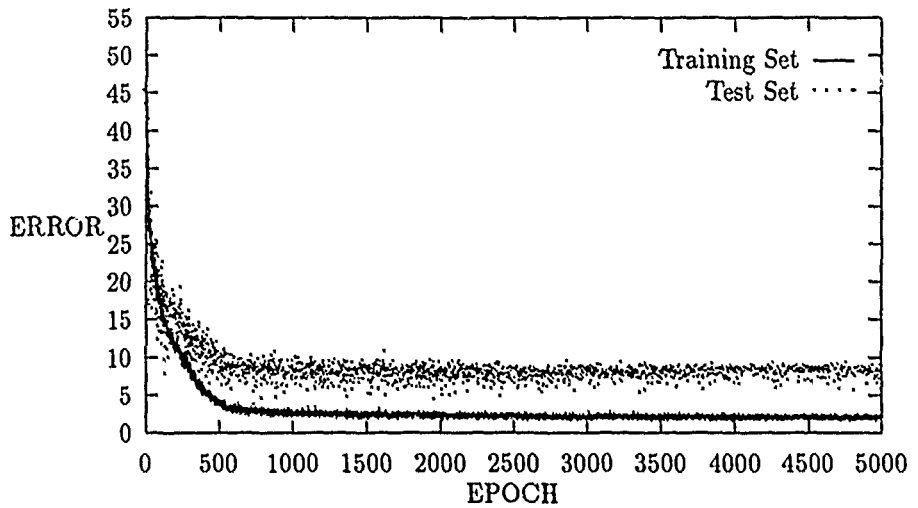


Figure 17. API Projectile Data – Classification Error for Optimal Structure

5.3.2 Application of Logistic Regression As a comparison to the results obtained with the multilayer perceptron, the SAS LOGISTIC procedure was applied to this application of classifying API projectiles. The confusion matrix for this procedure is shown in Figure 18.

The logistic regression methodology for this specific application may have been affected by the nonnormality of the API projectile data. Also, the logistic regression technique works well only if the groups are linearly separable. In the case of complete functionings and other types of functionings, an examination of the data shows that these cases overlap and are similar for all of the feature inputs.

Conclusion: The logistic regression technique for classifying API projectiles as 'complete' functions and 'other' does not outperform the multilayer perceptron.

5.3.3 Application of K-Nearest-Neighbor. The k-nearest-neighbor classifier was also applied to the API projectile data to determine if it could surpass the classification capabilities of the multilayer perceptron. This method was attempted for $k = 1$ through $k = 9$. (See Table 19.) The nearest-neighbor ($k = 1$) form of the

Table 18. API Projectile – Logistic Regression Confusion Matrix

TRAINING SET			
	Classified Complete	Classified Other	Total
True Complete	46.3% 25	53.7% 29	100% 54
True Other	6.4% 12	93.6% 175	100% 187
Total	37	204	241
Total Training Set Error Rate: .1411			

Table 19. API Projectile Data – Values of k for k-Nearest Neighbor Technique

k	Resubstitution Error Rate	Crossvalidation Error Rate	Validation Error Rate
1	0.00%	8.30%	8.30%
3	4.56%	8.71%	13.49%
5	5.39%	10.37%	19.50%
7	7.47%	12.03%	17.22%
9	9.54%	14.52%	20.95%

algorithm produced the smallest resubstitution, crossvalidation and validation error rates. The *average* confusion matrices for this k-nearest-neighbor method ($k = 1$) are shown in Table 20.

Conclusion: The nearest-neighbor classification technique for classifying API projectiles yielded an average error rate of 5.25% on the validation set and did not outperform the multilayer perceptron.

5.4 Feature Selection

5.4.1 Multilayer Perceptron Features.

Table 20. API Projectile Data – K-Nearest-Neighbor Confusion Matrices (Average Over 10 Runs)

RESUBSTITUTION			
	Classified Complete	Classified Other	Total
True Complete	100% 51.7	0% 0	100% 51.7
True Other	0% 0	100% 189.3	100% 189.2
Total	51.7	189.3	241

CROSSVALIDATION			
	Classified Complete	Classified Other	Total
True Complete	90.91% 47	9.09% 4.7	100% 51.7
True Other	1.90% 3.6	98.10% 185.7	100% 189.3
Total	50.6	190.4	241

VALIDATION			
	Classified Complete	Classified Other	Total
True Complete	90.24% 7.4	9.76% .8	100% 8.2
True Other	4.09% 1.3	95.91% 30.5	100% 31.8
Total	8.7	31.3	40
Total Validation Set Error Rate: 0.0525			

Table 21. API Projectile Data – Saliency Measures

	Saliency I		Saliency II	
Order of Significance	Feature	Measure	Feature	Measure
1	VAR 4 – SECT	19.44224	VAR 4 – SECT	504.616
2	VAR 1 – PLY	8.083079	VAR 1 – PLY	365.419
3	VAR 2 – VS	6.854682	VAR 2 –VS	159.026
4	VAR 3 – MS	1.44831	NOISE	38.613
5	NOISE	1.10228	VAR 3 – MS	35.387

5.4.1.1 Saliency. In this stage of the analysis, a uniform (0,1) noise variable was added as a feature input to aid in the determination of significant features. In the following discussion, the saliency measure according to Ruck will be termed "saliency I" and Tarr's measure will be called "saliency II." (23)(28) Table 21 shows the two saliency measures and the order of importance according to each measure. Each saliency measure was averaged over ten runs of the optimal structure multilayer perceptron.

Saliency II rated M_S (VAR 3) as less significant than noise and so, according to the procedure for determining significant features, this measure would immediately exclude M_S. Saliency I, on the other hand, left open the question as to whether M_S was a significant feature. Therefore, the procedure described in Chapter 3 was performed using saliency I. For the specific application of classifying API projectiles, the procedure was as follows:

1. In addition to PLY, VS, MS, and SECT, a noise feature, uniform (0,1), was included as input to the multilayer perceptron.
2. The network was trained repeadtedly in accordance with the established procedure for determining the optimal network structure.
3. Sixty training cycles were run with the optimal network structure. For each of these 60 runs, the saliency of each of the 5 feature inputs was recorded.

Table 22. API Projectile Data – Percentiles of the Saliency of Noise

Percentile	Critical Value
75th Percentile	1.3618
80th Percentile	1.4794
85th Percentile	1.6293
90th Percentile	1.8397
95th Percentile	2.0259
Distribution Mean: 1.10228 Distribution Standard Deviation: 0.581468	

4. From the 60 sample data points, the distribution of the saliency of noise was estimated. A histogram of the data and the resultant estimated distribution are graphed in Figure 18. The 75th through 95th percentiles of the resulting distribution are shown in Table 22. The 95th percentile was chosen as the required level of accuracy.

5. Figure 19 shows the relative position of the average saliency for each of the feature inputs. The average value of VAR 3 (M_S) was well below the 95th percentile of the saliency of noise.

   To ensure that this feature was well within the 95th percentile, a confidence interval for the mean saliency of VAR 3 was constructed. The saliency of noise distribution and the confidence interval for M_s are shown in Figure 20. The entire confidence interval for M_s falls well within the distribution of noise and, therefore, this feature was deleted from the set of inputs

6. The network was retrained and the resulting output error and classification error are shown in Figures 21 through 23.

An examination of the output error on the test set for the original four feature inputs versus the salient three feature set revealed a decrease in the training time and a slight decrease in output error. The output error curve also decreased at a

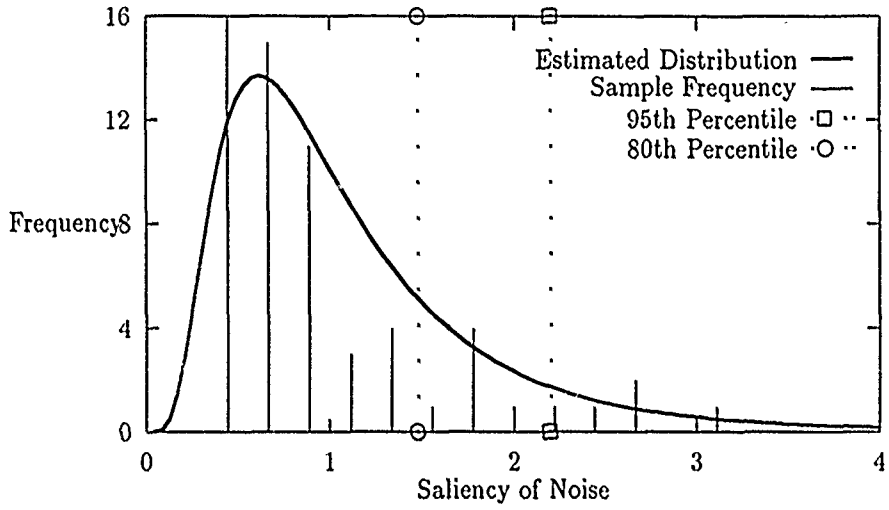


Figure 18. API Projectile Data – Estimated Distribution of Saliency of Noise

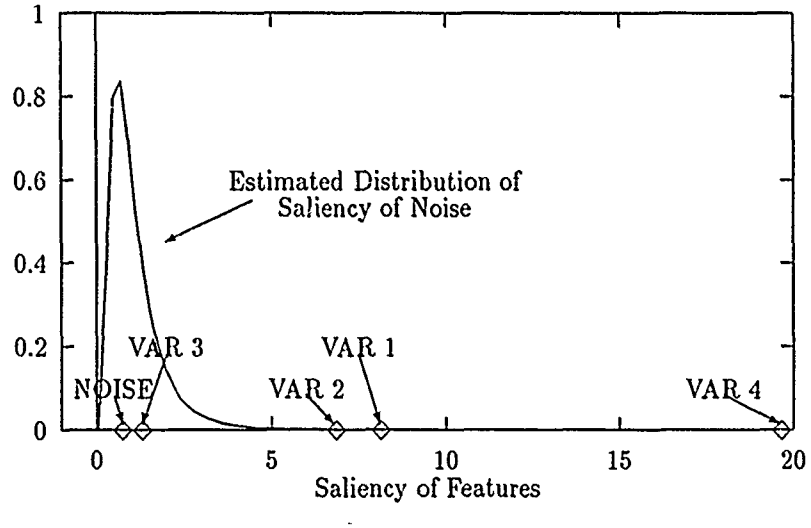


Figure 19. API Projectile Data – Saliency of All Features

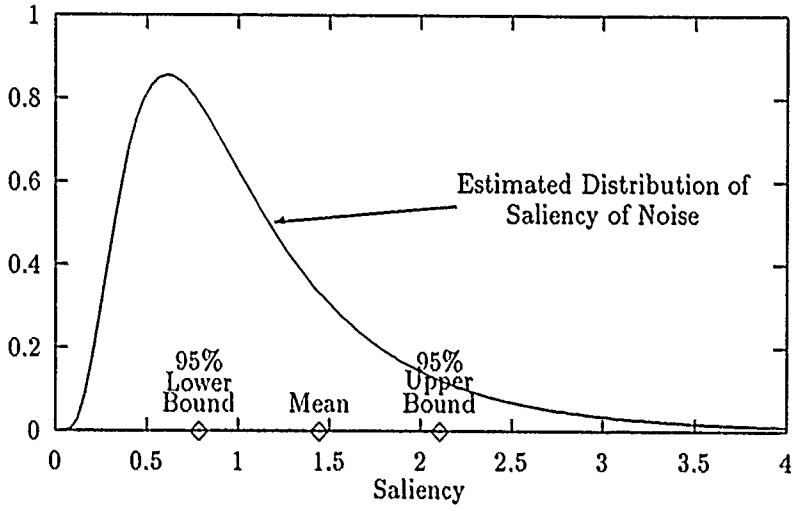


Figure 20. API Projectile Data – Confidence Interval for M_s Mean Saliency

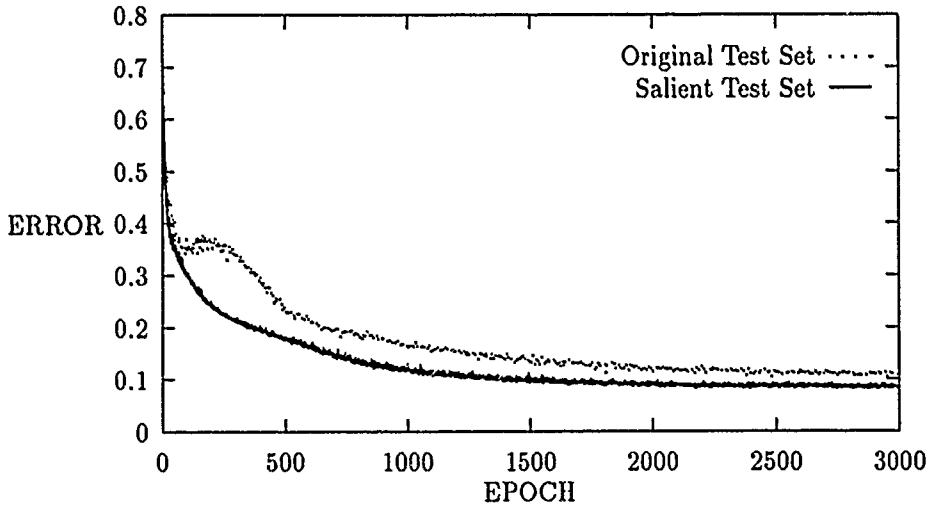


Figure 21. API Projectile Data – Output Error for Original Features vs Salient Features

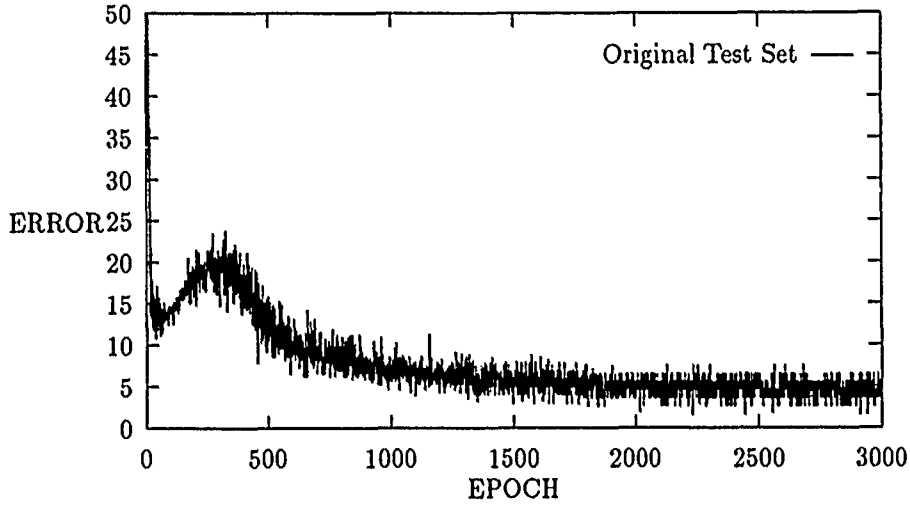


Figure 22. API Projectile Data – Classification Error for Original Features

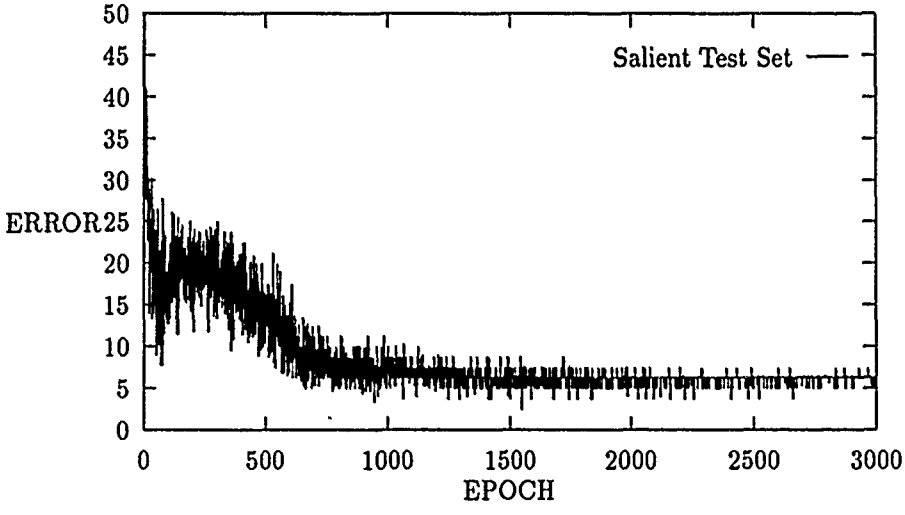


Figure 23. API Projectile Data – Classification Error for Salient Features

smoother more constant rate. The classification error on the test set for the salient feature set is no less than the original feature set, however, the error stabilizes more quickly. For this application, the test set was fairly small (41 feature vectors). With a larger test set, it is possible that the classification error would have decreased also.

The success of this procedure to determine significant features is not completely evident due to the small number of original input features. When the number of features causes the training time to be excessive, or causes the error rate to be high, the selection of significant features becomes especially important.

A word of caution is necessary. M_s, as one of the measures of the characteristic of a projectile, has been shown to be significant (15). What has been shown here, is that for the particular problem of classifying projectiles as 'complete' or 'other', M_s was not necessary.

Conclusion: Including a noise term as a feature input allows for an estimation of the distribution of the saliency of noise given the structure of the perceptron and the current candidate features. By examining the saliency of the other features in relation to this distribution, the insignificant terms can be deleted. Specifically, for API projectiles, it was determined that the feature M_S was not significantly different from noise and it was deleted. The resulting salient multilayer perceptron trained quicker, had a smaller output error, and the classification error stabilized sooner.

5.4.1.2 High-Order Inputs. In order to determine if second-order inputs would produce a more efficient classifier, the second-order correlation matrix was calculated. Tables 23 and 24 show the matrices for the output node associated with a complete function and for other types of functions. High absolute values in the ith, jth position of these matrices would indicate a high correlation with the second-

Table 23. Correlation Between Second-Order Terms and Ouput Node 1 (Complete Function)

	Feature 1	Feature 2	Feature 3	Feature 4
Feature 1	-0.0248	0.0044	-0.0115	-0.0000
Feature 2	-0.0121	-0.1516	-0.1216	-0.0000
Feature 3	-0.0092	0.8589	-0.1381	-0.0009
Feature 4	0.0012	0.4520	-0.0584	-0.0466

Table 24. Correlation Between Second-Order Terms and Ouput Node 2 (Other Function)

	Feature 1	Feature 2	Feature 3	Feature 4
Feature 1	-0.0300	0.0109	-0.0000	-0.0000
Feature 2	-0.0025	-0.0626	-0.1617	0.0000
Feature 3	0.0086	-0.1602	-0.1855	-0.0011
Feature 4	0.0020	0.0000	-0.0011	-0.0074

order term associated with the ith term times the jth term and the output of the perceptron.

The second-order terms M_sV_s and V_sSECT were most highly correlated with the network. The feature M_s was evaluated as contributing no more than noise to the discrimination of the classes. The high-correlation of the term M_sV_s with the ouput suggests that the term V_s is highly significant itself and we are observing the high first-order correlation of V_s with the output rather than a high correlation for the second-order term. The inclusion of the term V_sSECT looked more promising.

To test the use of this second-order term, a multilayer perceptron was constructed with only two inputs—V_sSECT and PLY. The ouput error for the resulting network is shown in Figure 24 and is compared the output error for the salient feature set. Notice that the high-order feature set causes the network to reach its minimum error at a slightly faster rate. The output error decreases to nearly the

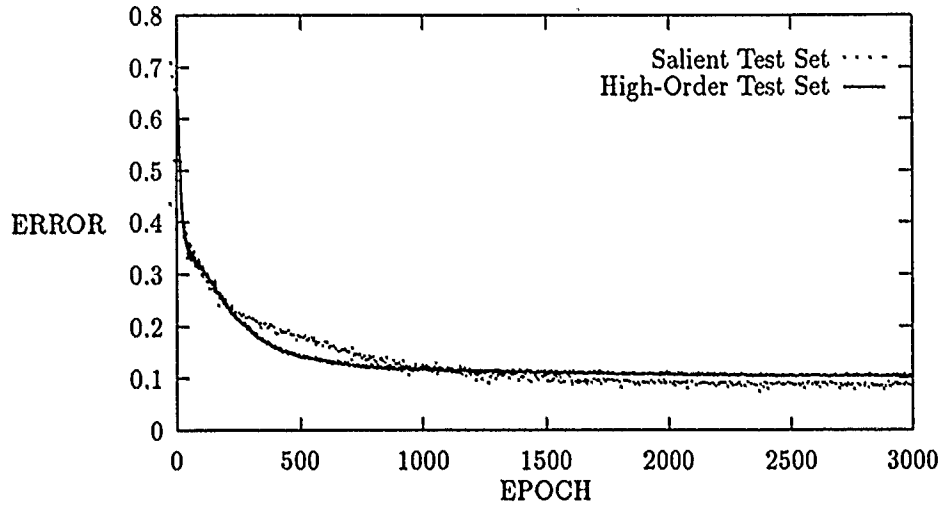


Figure 24. API Projectile Data - Output Error for High-Order Feature Set vs Salient Feature Set

same minimum as the salient feature set. The corresponding classification errors are shown in Figures 25 for the salient feature set and Figure 26 for the high-order feature set. Notice that with two terms, the classification error does not vary as greatly as the three feature salient set. Also, the high-order feature set never had a classification error rate of less than approximately 7.5% after 1,300 epochs while the salient feature set had rates below 7.5% throughout the 3000 epochs shown. Overall, both perceptrons converged to approximately the same classification error at 3000 epochs.

By constructing this second-order model, the set of features required for classification was reduced without the classification error increasing. Granted, the same amount of information must be collected for both networks (V_s, PLY, and $SECT$), however, the high-order perceptron required only two coefficients to describe the line that separates the feature vectors in the two-dimensional feature space. This translated to a reduced number of calculations and easier classification for new observations.

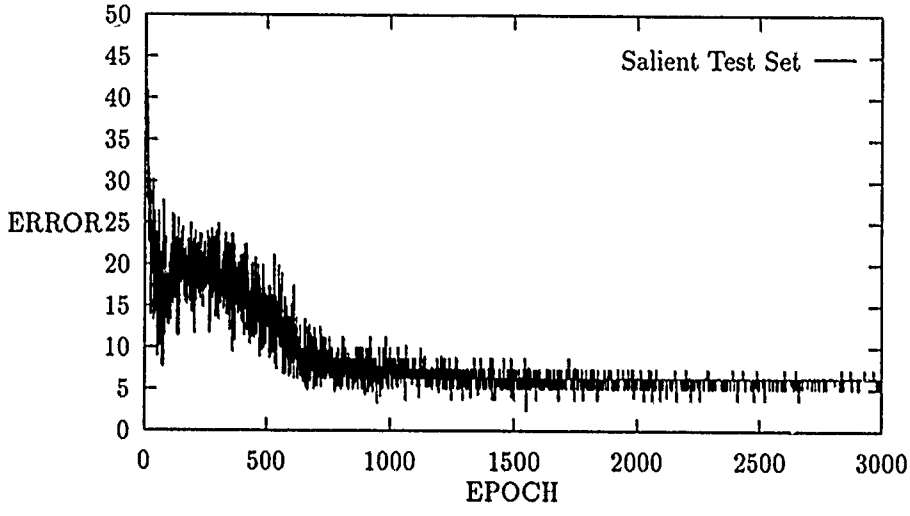


Figure 25. API Projectile Data – Classification Error for Salient Feature Set

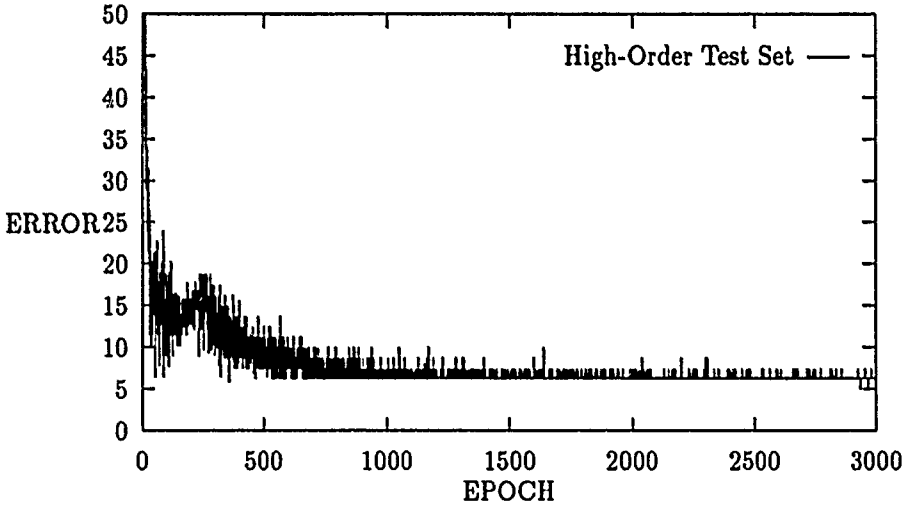


Figure 26. API Projectile Data – Classification Error for High-Order Feature Set

Since the high-order multilayer perceptron has only two inputs, it was possible to plot the features and their known classifications. Figure 27 shows the input feature PLY on the x-axis and V_sSECT on the y-axis. From this plot, it was evident that in the current feature set, the groups were approximately it linearly separable. Therefore, the inclusion of V_sSECT appeared to produce a feature space that was almost trivial to separate into "complete" and "other."

The use of high-order terms for this application led to the discovery of a set of features in which the feature space was easily separated. This application began with only four features. For an application with many features, and many data vectors, a method of finding candidate high-order terms could be very useful for both reducing the size of the network and determining an easily separable feature space. For example, the application of high-order terms would have been more relevant in the pilot classification application since so many inputs were coded as "1" and "-1."

Conclusion: Second-order inputs produced a multilayer perceptron with only two inputs, resulting in a linearly separable feature space whose discriminant function can be expressed with only two weights. If only the first-order terms had been used, this easily separable feature space would not have been discovered.

5.4.2 Logistic Regression Feature Selection Results. In order to compare the multilayer perceptron procedure for selecting significant features, with a more traditional method, the SAS LOGISTIC stepwise procedure was examined. Both the parameter estimates and their Chi-Square scores in the selection procedure serve as indicators of the importance of a feature. Table 25 shows the standardized parameter estimates and the Chi-Square scores for each of the input features including noise.

The results of logistic regression are very similar to those obtained using the saliency measures. When a stepwise selection criteria was introduced (at the .3 significance level), the feature M_S was deleted as insignificant. Notice also that

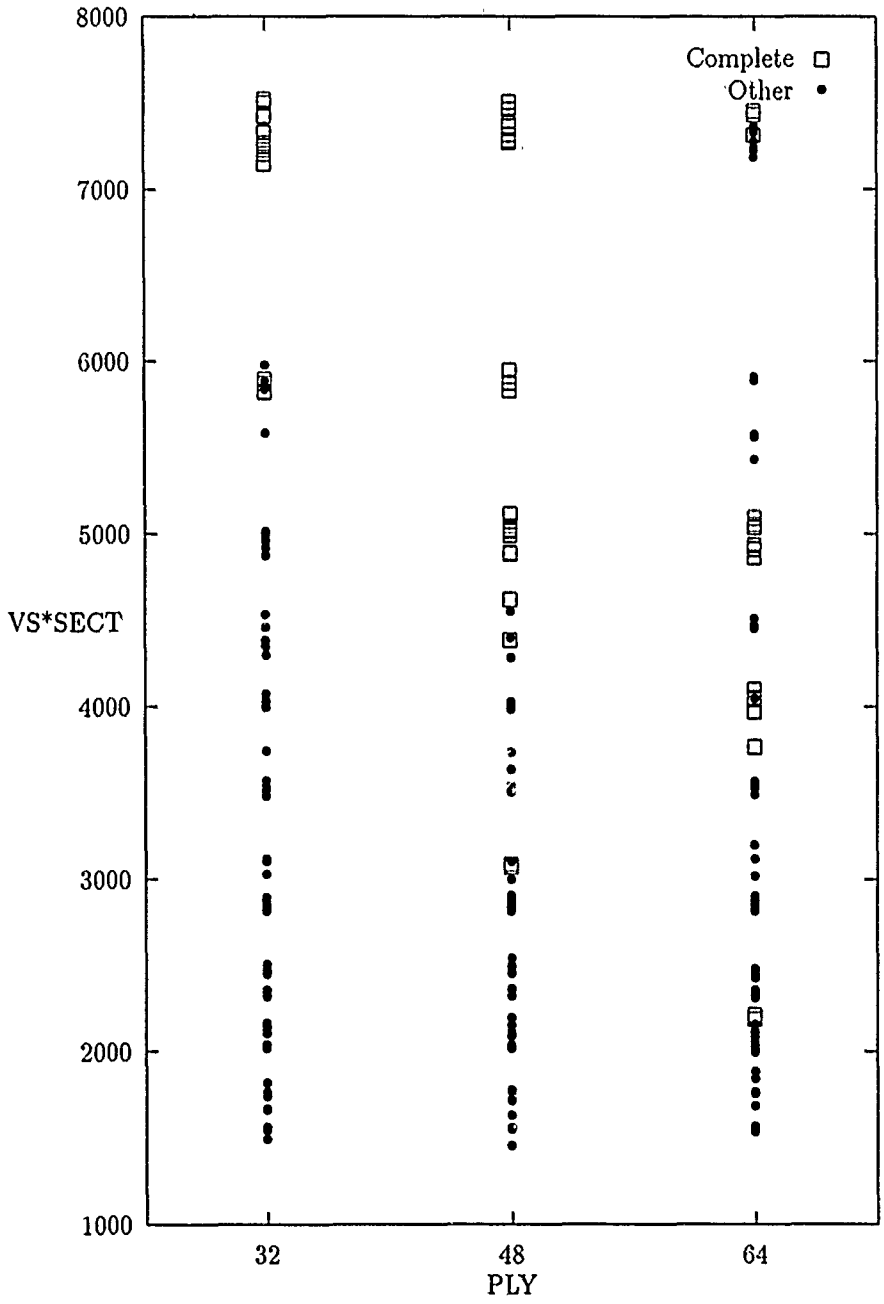


Figure 27. API Projectile Data – Plot of V_sSECT vs PLY

Table 25. API Projectile Data – Logistic Regression Stepwise Results

FEATURE NUMBER	FEATURE NAME	STANDARDIZED PARAMETER ESTIMATES	CHI-SQUARE SCORE	ORDER OF SIGNIFICANCE
VAR 1	PLY	-0.211756	3.5125	3
VAR 2	VS	-0.592804	18.6051	2
VAR 3	MS	-0.033271	0.0869	5
VAR 4	SECT	-0.870241	47.4262	1
VAR 5	NOISE	-0.085831	0.5991	4

the order of significance for the logistic regression function is the same as the order found using the saliency II measure. The STEPDISC procedure and the discriminant loadings produced similar results.

Conclusion: Comparable results were obtained with the multilayer perceptron feature selection techniques and the logistic regression stepwise feature selection technique.

5.5 Comparison of Methods

For this application, the run time for the multilayer perceptron classifier was less than the run time for the multivariate techniques. Run time was not a major concern for this application, since all runs took no more than one hour to complete.

Overall, the techniques attempted for this application were easy to complete compared to the pilot retention application. The multilayer perceptron technique was especially well suited the continuous data used in this application. Whether the perceptron can discriminate between high-dimensional binary variables as in the pliot application is still questionable.

Conclusion: Given the multilayer perceptron software and SAS, all classification methods were easy to use.

5.6 Application of Classifiers

In order to produce a classifier that can be used to SEB to predict the classification of other firings, a multilayer perceptron was trained with all 281 data points with the three salient features. The resulting weights are shown in Appendix G.

VI. Final Conclusions and Recommendations

6.1 Final Conclusions for Multilayer Perceptron Methods

- It was not determined whether decomposing categorical data elements into binary words for input to a multilayer perceptron is a viable technique.
- Determining the optimal architecture of a multilayer perceptron is facilitated by a predefined procedure, however, successive attempts to train the perceptron are still required. In addition, interactive synergism is ignored unless the interactions between the parameters (number of middle nodes, learning rate, etc.) are considered.
- Without a trained multilayer perceptron, it was not possible to use established saliency measures for feature reduction. Also, it is was possible to evaluate high-order terms for inclusion as inputs. Statistical classifiers, on the other hand, allow statistical tests for significance before the classifier has been designed.
- The introduction of noise as a feature input appears to present a method of determining the significance of a set of features by comparing their saliency to the saliency of the injected noise.
  - Several sample measures of the saliency of noise were obtained by successive runs of a multilayer perceptron.
  - The distribution of the saliency of noise was estimated.
  - Original feature inputs were examined to see if they exhibited saliencies less than that of noise.
  - Confidence intervals were constructed for features that were candidates for removal from the feature set. These confidence intervals were used to insure to the 95 percent level that the features being deleted had saliences no larger than noise.

- Appropriate features were deleted and the new salient feature set was used to retrain the perceptron.
- For one specific application, the multilayer perceptron trained quicker, had a lower output error and a more stable classification error with the salient feature set.

- The two saliency measures examined rank-ordered features similarly, however, the measurement scales and the relative measures were different.
- Multilayer perceptrons and the k-nearest-neighbor technique appear to yield similar error rates.
- Examining the correlations between high-order terms and the outputs of the multilayer perceptron may lead to a feature set with a quicker training time. In addition, it may be possible through these high-order terms to determine a feature set which easily separates the feature space.

6.2 Final Conclusions for Application 1 – Predicting Pilot's Retention Decisions

- The multilayer perceptron with 65 features as inputs could not be trained to discriminate between pilots that will leave and those that will remain in the Air Force.
- The logistic regression methodology using 65 features produced a classifier with an error rate of 19.0% for the FY 88 data set and 23.6% for the FY 89 data set. The classifier performed poorly when classifying those individuals that left the Air Force.
- The k-nearest-neighbor algorithm using 65 features produced a classifier with a total error of 20.67% for the FY 88 validation set and 24.87% for the FY 89 validation set. These error rates are likely to be unacceptable considering the poor performance classifying those that left the Air Force.

- Without a trained multilayer perceptron, the number of features cannot be reduced by saliency methods.
- Feature selection using parametric methods (logistic regression and stepwise discriminant analysis) produced a set of significant features, although the reduced feature set did not reduce the error rates.
- After all classification methods had been tested, the multilayer perceptron exhibited the least classification error rate, however, the error that was observed was significant.
- The multilayer perceptron technique, with the current computer software, was too computationally complex to produce a solution in a reasonable amount of time. The methods performed with SAS were computationally more efficient.

6.3 Final Conclusions for Application 2 – Predicting API Projectile Performance

- The multilayer perceptron classified the data in the validation set with an error rate of 4.5%.
- The logistic regression technique for classifying API projectiles as 'complete' functions and 'other' does not outperform the multilayer perceptron.
- The nearest-neighbor classification technique for classifying API projectiles yielded an average error rate of 5.5% on the validation set and did not on the average outperform the multilayer perceptron.
- Including a noise term as a feature input allows for an estimation of the distribution of the saliency of noise. By examining the saliency of the other features in relation to this distribution, the insignificant terms can be deleted. Specifically, for API projectiles, it was determined that the feature V_S was not significantly different from noise and it was deleted. The resulting salient multilayer

perceptron trained quicker, had a smaller output error, and the classification error stabilized with fewer epochs.

- Second-order inputs produced a more efficient classifier.
- Comparable results were obtained with the multilayer perceptron feature selection techniques and the logistic regression stepwise feature selection technique.

6.4 Recommendations for Multilayer Perceptron Technique

6.4.1 Distribution of the Saliency of Noise For the specific application of classifying API projectiles, it was shown that a formal procedure for determining significant features reduced training time and error. Now that this procedure has been demonstrated for a single case, it is necessary to show that the procedure is correct in general. A derivation of this nature would initially require a characterization of the distribution of the activations of the multilayer perceptron itself. The saliency I measure for example, is merely the sum of output and hidden layer activations and therefore, the underlying distribution of the saliency of a feature would be dependent on the distribution of the activations.

There is a certain intuitive appeal to the use of the lognormal distribution as a characterization of the saliency of a uniform random variate when a sigmoidal nonlinearity is used. However, the fact that the distribution of the output of a multilayer perceptron for a uniform input is lognormal has not been shown.

It is possible that one of the two saliency measures discussed in this analysis may be more meaningful for the procedure established to determine significant features. Saliency I was used in this analysis, however, perhaps saliency II gives a more realistic representation of the significance of a feature. Also, to this point, analysts have only been able to compare saliency measures between variables trained in the same network. Is there some optimal saliency measure for a feature that fully explains the discrimination between classes? What is needed is a formal comparison of the two saliency measures with other more established feature selection techniques,

in general terms. As with almost all neural network topics, it may be found that the distribution of saliency, the accuracy of the saliency measures and the difference between the measures are all problem/application specific.

6.4.2 Methods for Determining Optimal Architecture. In this analysis, what was called an "optimal" multilayer perceptron structure was really sub-optimal. The problem with a procedure that is optimal that all combinations of all parameters must be tested over the entire parameter space. This method can be too time consuming to be practical. In addition, once the optimal structure has been found for a single random selection of training, test and validation sets, what happens when new randomly chosen data sets are introduced?

One way that this problem can be addressed is by examining the interaction of the parameters. For example, if the learning rate is reduced, are more hidden nodes necessary? If so, it may be possible to test some subset of all possible parameter combinations. Perhaps, since it appears that the architecture of the multilayer perceptron is problem-dependent, the set of all multilayer perceptron applications could be segmented into classes and each of these classes examined in terms of the optimal parameter set. For example, the pilot retention classification problem had many input variables and binary data while the API projectile classification problem had few inputs with continuous data. These are two very different types of problems and it is possible that the selection of their structures is two very different problems.

6.5 Recommendations for Application 1 – Predicting Pilot's Retention Decisions

6.5.1 Unavailable Data As previously mentioned, not all of the variables which enter into each individual's decision-making process are available as data. Below are variables that it is felt contribute to the retention decision made by an Air

Force pilot, however, were not available for this analysis. If the information below was available, the two groups might be separable in the resulting feature space.

1. **Wife's Occupation** There is no reliable data available on the occupations of wives of Air Force pilots. Several surveys of Air Force wives have been conducted in the past, however, only summary statistics are available and specific occupations were not tracked.
2. **Number of Permanent Change of Station (PCS) Moves** It is probable that the number of PCS moves that a pilot and his family must undergo would have a significant bearing on a decision to leave military service. Currently, the personnel system does not maintain an accurate count of the number of moves that an individual makes during his career.
3. **Promotion Opportunities** Statistics on promotion opportunities are not currently available at the individual level. In addition, a variable representing promotion opportunity may be redundant in terms of this analysis since most of the factors that represent the probability of an individual's promotion may already be included.
4. **Taste for Military Life** No data was available for this analysis in either summary or individual form on the taste of Air Force pilots for military life. In the future, information from individual surveys could be recorded to quantify this variable.
5. **Adequacy of Housing** The adequacy of both on and off base housing may have an effect on an individual's retention decision. Currently, the lack of individual financial data available in the personnel system makes analysis of this information currently impossible.
6. **Perceived Next Assignment** Where the pilot thinks he will be assigned next and whether he will be flying in that assignment may be more important

than his current location. Obviously, until the assignment is actually made, personnel data files would not contain this information.

6.5.2 Economic Considerations. This analysis addressed the individual characteristics of Air Force pilots as a method to predict their retention decisions. However, there are other outside factors that enter into this decision. Unemployment rates, civilian airline hires, Air Force policies, etc. may cause individuals to leave the military without regard to their personal attributes. This analysis effort should represent only one variable in a function describing how *all* factors affect individual Air Force members and their retention decisions. Other economic variables such as the unemployment rate and military/civilian pay ration would represent the economic environment that the individual makes his decision in.

6.5.3 Decomposition of the Data Sets. Both the FY 88 and FY 89 data sets for this application were large (approximately 10,000 vectors). It is possible that there are really more than two groups underlying the data. Often, pilots who fly different types of aircraft have different motivations to leave or remain in military service. It seems that these major divisions of the pilots such as weapon system, could be used to decompose the data into smaller sets and then use discrimination techniques to classify the pilots as staying or leaving within these smaller sets.

It also might be possible to examine the principal components of the pilots in the data sets to determine if there are underlying characteristics that could be used rather than using all 65 feature inputs. This type of analysis would be performed by analyzing the principal components as described by Dillon and Goldstein (5:37).

Factor analysis could also be applied with the factor scores used as a method to cluster the individual pilots and then train a discriminator to classify pilots as "stay" or "leave" within these clusters.

6.5.4 Categorical Data Analysis Dillon and Goldstein list two difficulties associated with using standard data analysis techniques to analyze categorical data:

1. The dependent measure is not normally distributed.
2. The dependent measure does not display constant variance across the dummy-coded predictor variables (5:306).

There are methodologies available for analyzing categorical data that were not attempted in this research effort. Multidimensional contingency tables cross-classify the categories of one feature with the categories of another feature and can be used for data analysis. Loglinear models, for example, are designed to study the intercorrelations between categorical variables that form a contingency table. From this form of an analysis, it is possible to achieve a parsimonious description of the data in the form of a math model to account for the observations (5:303-336).

6.5.5 Application of Multilayer Perceptrons. Training Time. Although the multilayer perceptron never appeared to train on the data, it is possible that the number of epochs was too small to allow for convergence. The problem is that increasing the number of training epochs also increases the training time to unacceptable levels. (One run for the pilot problem which included all 65 variables ran for 2 weeks on a dedicated machine.) One way to speed training, is to change the computer code for the multilayer perceptron so that calculations are handled more efficiently and the perceptron can be trained in a shorter amount of time. The other way to speed training would be to find a faster computer if the resources are available.

High-Order Terms In his article on high-order inputs, Giles lists several ways to determine which high-order terms should be included as inputs to the multilayer perceptron (9). One method he considers is adding high-order iteratively to determine if training is affected. This would be a "trial-and-error" process which could be very time consuming. In addition, Giles mentions that the analysts knowledge of the problem may be used to determine which features should be included.

Giles specifically describes the situation where the inputs are binary with inputs of "1" and "-1." (9:4973) It seems that high order terms should be able to contribute positively to the problem of training experienced with the pilot classification problem. Although it appears that subjective methods will be needed to determine which feature to include.

6.6 Recommendations for Application 2 – Predicting API Projectile Performance

The results of the classification of API projectiles suggests that multilayer perceptrons could be a useful tool. Further study is necessary to ensure that the optimal structure of the perceptron was discovered and that the error rates cited are correct. The error rates for this analysis were based on 10 runs of the multilayer perceptron. This is probably not a sufficient number of runs for the levels of significance acceptable to the SEB. In addition, the classification problem for this analysis was a two-group problem, however, there are six classifications of API projectile firings. Other combinations of groups should be tested to find the discrimination problem that best suits the needs of the SEB.

Appendix A. *Random Data Configuration Program*

```
************************************************************************
*  RANDOM DATA CONFIGUARATION PROGRAM
*  Capt Lisa M. Belue
*
*  Date Last Modified: 29 February 1992
*
*  Program: datfig.f
*
*  Purpose:  The purpose of this code is to separate a data set into
*            the following three sets for use in classifiers:
*
*                 (1)  Training Set
*                 (2)  Test Set
*                 (3)  Validation Set
*
*  Important Variables:
*                 XTR:  Array of vectors randomly selected for the
*                       training set.
*                 XTS:  Array of vectors randomly selected for the
*                       test set.
*                 XVL:  Array of vectors randomly selected for the
*                       validation set.
*                 NR1:  Number of vectors in the data set
*                 N1TRAIN:  Number of vectors in training set
*                 N1TEST:  Number of vectors in test set
*                 N1VAL:  Number of vectors in validation set
*
************************************************************************

      EXTERNAL RNUNF
      INTRINSIC REAL,NINT

      INTEGER NR1,NCOL,N1COL,N1TRAIN,N1TEST,N1VAL,I

      PARAMETER(NCOL=6,N1COL=5,NR1=281,
     ;N1TRAIN=200,N1TEST=41,N1VAL=40)

      INTEGER CHOICE(NR1)

      REAL NOISE,X(NR1,NCOL),XTR(N1TRAIN,NCOL),
     ;XTS(N1TEST,NCOL),XVL(N1VAL,NCOL)

      OPEN(UNIT=11,FILE='shots1.dat',STATUS='UNKNOWN')
      OPEN(UNIT=13,FILE='train2.dat',STATUS='UNKNOWN')
      OPEN(UNIT=14,FILE='test2.dat',STATUS='UNKNOWN')
      OPEN(UNIT=15,FILE='val2.dat',STATUS='UNKNOWN')
```

```
      DO  10 I=1,NR1
       READ(11,*)(X(I,J), J=1,NCOL)
       write(*,*)(x(i,j),j=1,ncol)
 10   CONTINUE


***** THE FOLLOWING CODE RANDOMLY SELECTS VECTORS FOR THE TRAINING SET
***** AND WRITES THE VECTORS TO A FILE.

      DO 500 J=1,NR1
        CHOICE(J)=0
 500  CONTINUE

      DO 530 II =  1,N1TRAIN
      WRITE(*,*)'SELECTING TRAIN',II
  14   CONTINUE
      TEMP=RNUNF()
      JJ=NINT(TEMP*NR1)
      IF ((JJ.LE.NR1).AND.(JJ.GT.0)) THEN
        DO 510 K =1,NR1
          IF (JJ.EQ.CHOICE(K)) GO TO  14
 510    CONTINUE
        DO 520 KK=1,NCOL
          XTR(II,KK)=X(JJ,KK)
 520    CONTINUE
        CHOICE(II)=JJ
      ELSE
        GO TO  14
      END IF
 530  CONTINUE

      DO 540 I=1,N1TRAIN
        WRITE(*,*)'WRITING TRAIN',I
        NOISE=RNUNF()
        WRITE(13,*)XTR(I,1),XTR(I,2),XTR(I,3),XTR(I,4)
     ;,XTR(I,6)

 540  CONTINUE

***** THE FOLLOWING CODE RANDOMLY SELECTS VECTORS FOR THE TEST SET
***** AND WRITES THE VECTORS TO A FILE.

      DO 550 II=1,N1TEST
      WRITE(*,*)'SELECTING TEST',II
  15   CONTINUE
      TEMP=RNUNF()
      JJ=NINT(TEMP*NR1)
      IF ((JJ.LE.NR1).AND.(JJ.GT.0)) THEN
        DO 560 K=1,NR1
          IF (JJ.EQ.CHOICE(K)) GO TO  15
```

```
560     CONTINUE
        DO 570 KK =  1,NCOL
          XTS(II,KK)=X(JJ,KK)
570     CONTINUE
        CHOICE(II+N1TRAIN)=JJ
      ELSE
        GO TO  15
      END IF
550   CONTINUE

      DO 580 I =1,N1TEST
        WRITE(*,*)'WRITING TEST',I
        NOISE=RNUNF()
        WRITE(14,*)XTS(I,1),XTS(I,2),XTS(I,3),XTS(I,4)
     ;,XTS(I,6)

580   CONTINUE

***** THE FOLLOWING CODE READS THE VECTORS NOT SELECTED FOR THE TRAINING
***** OR TEST SET INTO A VALIDATION FILE.

      CNT=0
      DO 590 I=1,NR1
        DO 600 K=1,NR1
          IF (I.EQ.CHOICE(K)) GO TO 590
600     CONTINUE
      CNT=CNT+1
      WRITE(*,*)'SELECTING VALIDATION',CNT
      DO 610 KK=1,NCOL
        XVL(CNT,KK)=X(I,KK)
610   CONTINUE
590   CONTINUE

      DO 620 I=1,CNT
        WRITE(*,*)'WRITING VALIDATION',I
        NOISE=RNUNF()
      WRITE(15,*)XVL(I,1),XVL(I,2),XVL(I,3),XVL(I,4)
     ;,XVL(I,6)

620   CONTINUE

      END
***** END OF PROGRAM*****
```

Appendix B. *Multilayer Perceptron Program*

```
*************************************************************************
*  MULTILAYER PERCEPTRON (SINGLE HIDDEN LAYER)
*  Capt Lisa M. Belue
*
*  Date Last Modified: 29 February 1992
*
*  Program:  NN5.f
*
*  Purpose:  The purpose of this code is to train a multilayer perceptron
*            to classify groups of data based on training and test data
*            presented to the perceptron.  In addition, this code
*            determines the error on a validation set, calculates the
*            saliency of the inputs and constructs correlation matrices to
*            evaluate high-order inputs
*
*  Tools:    Artificial Neural Network Estimator
*            Characteristics:  Single Hidden Layer
*                              Two Layers of Weights
*                              Sigmoid Non-Linearity
*
*  Goal:  This particular code is used to determine the effectiveness
*         of the neural network classifier for a particular
*         application.
*
*   Main Program:  The main program consists of calls for five subroutines:
*                  (1)  INPUT:  Input the data in the correct format.
*                  (2)  NORMAL:   Normalize the feature vectors.
*                  (3)  ANN:  Run a single layer neural network.
*                  (4)  OUTPUT:  Output results and weights.
*                  (5)  SALIENCY:  Computes the saliency of the features.
*                  (6)  VALIDATE:  Classifies individuals and determines
*                                  number classified correctly.
*                  (7)  CORRELATE:  Computes the second-order correlation
*                                   of inputs to outputs.
*
*
*   Important Variables:
*            NE:  Number of epochs
*            NM:  Number of hidden nodes
*            TRAIN:  Number of vectors in the training set
*            TEST:  Number of vectors in the test set
*            TOL1:  Error tolerance for stopping
*            TOL2:  Iteration tolerance for stopping
*            C1:  Gain parameter
*            C2:  Momentum parameter
*            NGRP2:  Number of classification groups
```

```
*            F:  Number of features plus bias plus number of groups
*            NM1:  Number of hidden nodes plus one for bias
*            NVAR1:  Number of variables plus one for bias
*            IREPTR:  Number of exemplars in the training set
*            IREPTS:  Number of exemplars in the test set
*
*
*******************************************************************************
      PROGRAM NN5

      INTEGER TRAIN,TEST,IREPTR,IREPTS,IREPVL,F,NGRP2,
     ;NVAR,NVAR1,NE,NM,NM1,NDIV,NDIV1,NSAL,NEF

      REAL TOL1, TOL2, C1, C2

***** It is necessary to alter the values of the parameters below
***** to suit the specific application

      PARAMETER(NE=2,NM=20,TRAIN=200,TEST=41,
     ;IREPVL=40,TOL1=10.0,
     ;TOL2=10.0,C1=0.2,C2=0.0,NGRP2=2,NVAR=5,
     ;NM1=NM+1,NVAR1=NVAR+1,
     ;IREPTR=TRAIN,IREPTS=TEST,F=NVAR1+NGRP2,NDIV=10,
     ;NDIV1=NDIV+1,
     ;NSAL=NDIV1*TRAIN)

      REAL WW1(NVAR1,NM,3),WW2(NM1,NGRP2,3),XXERR(NE),
     ;YYERR(NE),CCONF(NE,NGRP2,NGRP2),CCCONFT(NE,NGRP2,NGRP2),
     ;MISS(NE),MISST(NE)

      REAL XTEMP(IREPTR,NVAR1), DTEMP(IREPTR,NGRP2),
     ;TRFEAT(IREPTR,NVAR1),TSFEAT(IREPTS,NVAR1)

      REAL XX1(IREPTR,NVAR1), X1(IREPTR,F),Y1(IREPTS,F),
     ;RANGE(NVAR),DD3(IREPTR,NGRP2),X2(NM1),X3(NGRP2),
     ;D3(IREPTR,NGRP2),Y2(NM1),Y3(NGRP2),E3(IREPTS,NGRP2),
     ;MAX(NVAR),MIN(NVAR)

      REAL X2SAL(NM1),X3SAL(NGRP2),XSAL(NSAL,NVAR1),
     ;XDIV(NVAR1,NDIV1),SAL(NVAR1)

      REAL V1(IREPVL,NVAR1),V2(NM1),V3(NGRP2),VAL(IREPVL,NVAR1)

      INTEGER CHOICE(IREPTR)

      REAL U2(NM1),U3(NGRP2),XX(IREPTR,F),YY(IREPTR,NGRP2)

***** Files for both input and output:

      OPEN(UNIT=11,FILE='out.dat',STATUS='UNKNOWN')
      OPEN(UNIT=23,FILE='train1.dat',STATUS='UNKNOWN')
```

```
OPEN(UNIT=15,FILE='test1.dat',STATUS='UNKNOWN')
OPEN(UNIT=21,FILE='weights.dat',STATUS='UNKNOWN')
OPEN(UNIT=19,FILE='errhis.dat',STATUS='UNKNOWN')
OPEN(UNIT=24,FILE='e.dat',STATUS='UNKNOWN')
OPEN(UNIT=20,FILE='confu.dat',STATUS='UNKNOWN')
OPEN(UNIT=22,FILE='sal.dat',STATUS='UNKNOWN')
OPEN(UNIT=16,FILE='val1.dat',STATUS='UNKNOWN')
OPEN(UNIT=17,FILE='valout.dat',STATUS='UNKNOWN')
OPEN(UNIT=18,FILE='correl.dat',STATUS='UNKNOWN')

WRITE(11,*) 'INVESTIGATION OF MULTILAYER PERCEPTRONS AND'
WRITE(11,*) 'MULTIVARIATE DISCRIMINANT ANALYSIS FOR'
WRITE(11,*) 'CLASSIFICATION AND PREDICTION'
WRITE(11,*)
WRITE(11,*) 'LISA M. BELUE'
WRITE(11,*)
WRITE(11,*) 'TRAINING SAMPLE SET = ',TRAIN
WRITE(11,*) 'TEST SAMPLE SET1 = ',TEST

WRITE(*,*)'ABOUT TO DO SUBROUTINE INPUT'

CALL INPUT(IREPTR,IREPTS,F,NVAR1,NVAR,X1,Y1,D3,E3,
;TRFEAT,TSFEAT,NGRP2)

WRITE(*,*)'ABOUT TO DO SUBROUTINE NORMAL'

CALL NORMAL(F,IREPTR,IREPTS,X1,Y1,NVAR,RANGE
;,MAX,MIN)

WRITE(*,*)'ABOUT TO DO ANN'

CALL ANN(IREPTR,IREPTS,NE,NGRP2,NVAR,NM,X1,
;X2,X3,D3,Y1,Y2,Y3,E3,WW1,WW2,TOL1,TOL2,C1,C2,
;XXERR,YYERR,CCONF,CCCONFT,NEF,MISS,MISST,NM1,NVAR1,XTEMP,
;DTEMP,DD3,XX1,CHOICE)

WRITE(*,*)'ALL DONE WITH ANN'


CALL OUTPUT(NM,NE,C1,C2,TRAIN,TEST,CCONF,NEF,
;CCCONFT,NGRP2)

WRITE(*,*)'ABOUT TO DO SALIENCY'

CALL SALIENCY(WW1,WW2,X1,IREPTR,IREPTS,NVAR1
;,NGRP2,NDIV,
;NDIV1,NSAL,NM1,NM,X2SAL,X3SAL,XSAL,XDIV,SAL)

WRITE(*,*)'ALL DONE WITH SALIENCY'

WRITE(*,*)'ABOUT TO DO VALIDATION'
```

```
      CALL VALID(V1,V2,V3,VAL,WW1,WW2,IREPVL,NVAR,NVAR1,NGRP2,
     ;NM,NM1,NE,RANGE,MAX,MIN,G3,VCONFT)

      WRITE(*,*)'ALL DONE WITH VALIDATION'

      WRITE(*,*)'ABOUT TO DO CORRELATE'

      CALL CORRELATE(X1,W1,W2,IREPTR,NVAR1,NM1,NM,NGRP2,
     ;U2,U3,NVAR,NM,XX,YY,YSUM,YBAR,XSUM,XBAR,COR2)

      WRITE(*,*)'ALL DONE WITH CORRELATE'

      WRITE(*,*)'***** ALL DONE WITH PROGRAM NN3 *****'

      END


*1111111111111111111111111111111111111111111111111111111111111111111111111111
*
* Subroutine INPUT:  This subroutine inputs the data to the feature
*                    matrices which are originally located in files
*                    TRPILOT.dat and TSPILOT.dat.
*
*     TRFEAT(I,J):  Array that contains the training vectors as read
*                   from the input files.
*                   I:  Vector number
*                   J:  Input variable number
*     TSFEAT(I,J):  Array that contains the test vectors as read
*                   from the input files.
*                   I:  Vector number
*                   J:  Input variable number
*
*1111111111111111111111111111111111111111111111111111111111111111111111111111

      SUBROUTINE INPUT(IREPTR,IREPTS,F,NVAR1,NVAR,
     ;X1,Y1,D3,E3, TRFEAT, TSFEAT,NGRP2)

      INTEGER F,NVAR,NVAR1,IREPTR,IREPTS,NGRP2

      REAL TRFEAT(IREPTR,NVAR1), TSFEAT(IREPTS,NVAR1),
     ;X1(IREPTR,F), Y1(IREPTS,F), D3(IREPTR,NGRP2), E3(IREPTS,NGRP2)

      DO 10 I=1,IREPTR
         READ(23,100)(TRFEAT(I,J), J=1,NVAR1)
         WRITE(*,*)'INPUT TRAINING VECT',I
  10  CONTINUE
      DO 20 I2=1,IREPTS
         READ(15,100)(TSFEAT(I2,J2), J2=1,NVAR1)
         WRITE(*,*)'INPUT TEST VECT',I2
  20  CONTINUE
```

```
***** It is necessary to change the format statement below to read
***** the data from the training, test, and validation data sets.

 100  FORMAT(F7.4,2X,F7.2,2X,F7.3,2X,F7.5,2X,F7.5,2X,F7.5)

      DO 30 I3=1,IREPTR
        IF (TRFEAT(I3,NVAR1).EQ.1) D3(I3,1)=1.0
        IF (TRFEAT(I3,NVAR1).EQ.1) D3(I3,2)=0.0
        IF (TRFEAT(I3,NVAR1).EQ.0) D3(I3,2)=1.0
        IF (TRFEAT(I3,NVAR1).EQ.0) D3(I3,1)=0.0
  30  CONTINUE

      DO 40 I4=1,IREPTS
        IF (TSFEAT(I4,NVAR1).EQ.1.0) E3(I4,1)=1.0
        IF (TSFEAT(I4,NVAR1).EQ.1.0) E3(I4,2)=0.0
        IF (TSFEAT(I4,NVAR1).EQ.0.0) E3(I4,2)=1.0
        IF (TSFEAT(I4,NVAR1).EQ.0.0) E3(I4,1)=0.0
  40  CONTINUE

      DO 70 I7=1,IREPTR
        DO 80 J3= 1,NVAR
          X1(I7,J3)=TRFEAT(I7,J3)
 80     CONTINUE
          X1(I7,NVAR1)=1.0
 70   CONTINUE

      DO 90 I8=1,IREPTS
        DO 110 J4= 1,NVAR
          Y1(I8,J4)=TSFEAT(I8,J4)
 110    CONTINUE
          Y1(I8,NVAR1)=1.0
 90   CONTINUE

      RETURN
      END


*1111111111111111111111111111111111111111111111111111111111111111111111111111111111
*
*  Subroutine NORMAL:  This subroutine normalizes each of the feature
*                      vectors to values between 0 and 1.
*
*     RANGE(J):  Array for the range of the jth input feature.
*     MAX(J):  The maximum value of input feature j.
*     MIN(J):  The minumum value of the input feature j.
*
*1111111111111111111111111111111111111111111111111111111111111111111111111111111111

      SUBROUTINE NORMAL(F,IREPTR,IREPTS,X1,Y1,NVAR,RANGE,
     ;MAX,MIN)
```

```
      INTEGER F,IREPTR,IREPTS,NVAR

      REAL X1(IREPTR,F),Y1(IREPTS,F),RANGE(NVAR),MAX(NVAR)
     ;,MIN(NVAR)

      DO 55 J=1,NVAR
      MAX(J)=-10.0
      MIN(J)=10000.0
        DO 10 I=1,IREPTR
          IF(X1(I,J).GT.MAX(J)) MAX(J)=X1(I,J)
          IF(X1(I,J).LT.MIN(J)) MIN(J)=X1(I,J)
10      CONTINUE

        DO 20 I2=1,IREPTS
          IF(Y1(I2,J).GT.MAX(J)) MAX(J)=Y1(I2,J)
          IF(Y1(I2,J).LT.MIN(J)) MIN(J)=Y1(I2,J)
20      CONTINUE
        RANGE(J)=MAX(J)-MIN(J)
        IF (RANGE(J).LT.0.001) RANGE(J)=0.001

        DO 30 I3=1,IREPTR
          IF (X1(I3,J).GE.0) X1(I3,J)=(X1(I3,J)-MIN(J))/RANGE(J)
          IF (X1(I3,J).LT.0) X1(I3,J)=(X1(I3,J)-ABS(MIN(J)))/RANGE(J)
          WRITE(*,*)'NORMAL TRAINING VECT',I3
30      CONTINUE
        DO 40 I4=1,IREPTS
          IF (Y1(I4,J).GE.0) Y1(I4,J)=(Y1(I4,J)-MIN(J))/RANGE(J)
          IF (Y1(I4,J).LT.0) Y1(I4,J)=(Y1(I4,J)-ABS(MIN(J)))/RANGE(J)
          WRITE(*,*)'NORMAL TEST VECT',I4
40      CONTINUE
55    CONTINUE

      RETURN
      END


*1111111111111111111111111111111111111111111111111111111111111111111111111111
*
*  Subroutine OUTPUT:  This subroutine outputs a report showing the
*                      classification of the test set.
*
*1111111111111111111111111111111111111111111111111111111111111111111111111111

      SUBROUTINE OUTPUT(NM,NE,C1,C2,TRAIN,TEST,CCONF,NEF,CCCONFT,NGRP2)

      REAL C1,C2,CCONF(NE,NGRP2,NGRP2),CCCONFT(NE,NGRP2,NGRP2)

      INTEGER NM,NE,TRAIN,TEST,NEF,NGRP2

      WRITE(11,*)
```

```
      WRITE(11,*) 'NUMBER OF MIDDLE NODES = ',NM
      WRITE(11,*) 'NUMBER OF EPOCHS = ',NE
      WRITE(11,*) 'C1 = ',C1,'  C2 = ',C2
      WRITE(11,*)
      WRITE(11,*) 'NOTE:  THIS IS THE TRAINING SET'

      WRITE(11,*) 'NUMBER OF TRAIN VECTORS = ',TRAIN
      WRITE(11,*)
      WRITE(11,*) '************************************'
      WRITE(11,*) '*         | |CLASSIFIED|CLASSIFIED*'
      WRITE(11,*) '*         | |CLASS 0   |CLASS 1   *'
      WRITE(11,*) '*_________|_|__________|__________*'
      WRITE(11,*) '*TRUE     | |          |          *'
      WRITE(11,700)'*CLASS 0  | |',CCONF(NEF,1,1),'%    |'
     ;,CCONF(NEF,1,2),'%    *'
      WRITE(11,*) '*_________|_|__________|__________*'
      WRITE(11,*) '*TRUE     | |          |          *'
      WRITE(11,700)'*CLASS 1  | |',CCONF(NEF,2,1),'%    |',
     ;CCONF(NEF,2,2),'%    *'
      WRITE(11,*) '************************************'
      WRITE(11,*)
      WRITE(11,*) 'NOTE:  THIS IS THE TEST SET'
      WRITE(11,*) 'NUMBER OF TEST VECTORS = ',TEST
      WRITE(11,*)
      WRITE(11,*) '************************************'
      WRITE(11,*) '*         | |CLASSIFIED|CLASSIFIED*'
      WRITE(11,*) '*         | |CLASS 0   |CLASS 1   *'
      WRITE(11,*) '*_________|_|__________|__________*'
      WRITE(11,*) '*TRUE     | |          |          *'
      WRITE(11,700)'*CLASS 0  | |',CCCONFT(NEF,1,1),'%    |'
     ;,CCCONFT(NEF,1,2),'%    *'
      WRITE(11,*) '*_________|_|__________|__________*'
      WRITE(11,*) '*TRUE     | |          |          *'
      WRITE(11,700)'*CLASS 1  | |',CCCONFT(NEF,2,1),'%    |',
     ;CCCONFT(NEF,2,2),'%    *'
      WRITE(11,*) '************************************'

  700 FORMAT(A12,2(F5.1,A6))
  710 FORMAT(1X,11(F6.3,1X))
  725 FORMAT(35(F9.6,1X))
  726 FORMAT(4(F3.1,1X))

      RETURN
      END


*22222222222222222222222222222222222222222222222222222222222222222222222222222222222222
*
* Subroutine ANN:  Trains and Tests an artificial neural network
*
*     D3:  Vector of Desired Outputs for training set
```

```
*     X1:  Vector of Input features for training set
*     X2:  Vector of Outputs of Hidden Layer after through sigmoid for training
*     X3:  Vector of Outputs of Output Layer after through sigmoid for training
*     W1(i,j,l):  Weight connecting input layer i to hidden layer j
*                 l=3 means weight corresponds to current iteration
*                 l=2 means weight corresponds to last iteration
*                 l=1 means weight corresponds to iteration before last
*     W2(j,k,l):  Weight connecting hidden layer j to output layer k
*                 l=3 means weight corresponds to current iteration
*                 l=2 means weight corresponds to last iteration
*                 l=1 means weight corresponds to iteration before last
*     TOL1:  Error tolerance for stopping
*     TOL2:  Iteration tolerance for stopping
*     NI:  Counter for number of input nodes
*     NM:  Counter for number of hidden nodes
*     NO:  Counter for number of output nodes
*     NXTR:  Number of exemplars in training set
*     NXTS:  Number of exemplars in test set
*     NE:  Number of epochs through training set
*     NEF:  Number of epochs to reach termination of training
*     C1:  Gain parameter
*     C2:  Momentum parameter
*     E3:  Vector of desired outputs for training set
*     Y1:  Vector of input features for test set
*     Y2:  Vector of outputs of Hidden Layer after through sigmoid
*          for test set
*     Y3:  Vector of Outputs of Output Layer after through sigmoid
*          for test set
*     XERR:  Training set error for each epoch
*     YERR:  Test set error for each epoch
*     CONF:  Confusion matrix for training set
*     CCONFT:  Confusion matrix for test set
*     MISS:  Percentage of incorrect classifications -- training set
*     MISST:  Percentage of incorrect classifications -- test set
*     XTEMP:  Temporary array for X1 array
*     DTEMP:  Temporary array for D3 array
*
* 22222222222222222222222222222222222222222222222222222222222222222222222222

      SUBROUTINE ANN(NXTR,NXTS,NE,NO,NI,NM,X1,
     ;X2,X3,D3,Y1,Y2,Y3,E3,W1,W2,TOL1,TOL2,C1,C2,XERR,YERR,CONF
     ;,CCONFT,
     ;NEF,MISS,MISST,NM1,NI1,XTEMP,DTEMP,DD3,XX1,CHOICE)

      INTEGER NXTR,NXTS,NE,NO,NI,NM,I,II,JJ,J,KK,K,LL,MM,
     ;NEF,ROW,COL,NI1,CHOICE(NXTR),NM1

      REAL X1(NXTR,NI1),X2(NM1),X3(NO),D3(NXTR,NO),Y1(NXTS,NI1),
     ;Y2(NM1),Y3(NO),E3(NXTS,NO),W1(NI1,NM,3),W2(NM1,NO,3),
     ;TOL1,TOL2,C1,C2,XERR(NE),YERR(NE),TEMP,TERM1,TERM2,MAX,
     ;CONF(NE,NO,NO),CCONFT(NE,NO,NO),MISS(NE),MISST(NE),
```

```
     ;XTEMP(NXTR,NI1),DTEMP(NXTR,NO),XX1(NXTR,NI1),DD3(NXTR,NO)

      EXTERNAL RNUNF
      INTRINSIC REAL,NINT

***** Write feature vectors and desired outputs to temporary files
      DO 1 I=1,NXTR
        DO 2 J=1,NI1
          XX1(I,J)=X1(I,J)
 2      CONTINUE
        DO 3 J=1,NO
          DD3(I,J)=D3(I,J)
 3      CONTINUE
 1    CONTINUE

***** Initailize weights to random numbers (normal distribution)
      DO 4 J=1,NM
        DO 5 I=1,NI1
          W1(I,J,1)=0.0
          W1(I,J,2)=RNUNF()
          W1(I,J,2)=W1(I,J,2)-.5
 5      CONTINUE
 4    CONTINUE
      DO 6 J=1,NM1
        DO 7 K=1,NO
          W2(J,K,1)=0.0
          W2(J,K,2)=RNUNF()
          W2(J,K,2)=W2(J,K,2)-.5
 7      CONTINUE
 6    CONTINUE

***** Initialize test and train confusion matrices
      DO 8 LL=1,NE
        DO 9 I=1,NO
          DO 10 J=1,NO
            CONF(LL,I,J)=0.0
            CCONFT(LL,I,J)=0.0
 10       CONTINUE
 9      CONTINUE
 8    CONTINUE


***** Begin training epochs
      DO 11 LL=1,NE
      WRITE(*,*)'EPOCH = ',LL
        XERR(LL)=0.0
        YERR(LL)=0.0

***** Randomly select training vectors for input to network
        DO 12 I=1,NXTR
          CHOICE(I)=0
```

```
 12     CONTINUE

        DO 13 II = 1,NXTR
 14     CONTINUE
        TEMP=RNUNF()
        JJ=NINT(TEMP*NXTR)
        IF ((JJ.LE.NXTR).AND.(JJ.GT.0)) THEN
          DO 15 K=1,NXTR
            IF (JJ.EQ.CHOICE(K)) GO TO 14
 15       CONTINUE
          DO 16 KK=1,NI1
            XTEMP(II,KK)=XX1(JJ,KK)
 16       CONTINUE
          DO 17 KK=1,NO
            DTEMP(II,KK)=DD3(JJ,KK)
 17       CONTINUE
          CHOICE(II)=JJ
        ELSE
          GO TO 14
        ENDIF
 13     CONTINUE

        DO 18 II=1,NXTR
          DO 19 JJ=1,NI1
            X1(II,JJ)=XTEMP(II,JJ)
 19       CONTINUE
          DO 20 JJ=1,NO
            D3(II,JJ)=DTEMP(II,JJ)
 20       CONTINUE
 18     CONTINUE
***** End of random selection for training vectors

***** Introduce training vectors to network
        DO 50 MM=1,NXTR

***** Hidden layer activation
          X2(NM1)=1.0
          DO 60 J=1,NM
            X2(J)=0.0
            DO 70 I=1,NI1
              X2(J)=X2(J)+X1(MM,I)*W1(I,J,2)
 70         CONTINUE
*           (Avoid over/under flows)
            IF (X2(J) .GT. 60.0) X2(J)=60.0
            IF (X2(J) .LT. -60.0) X2(J)=-60.0
            X2(J)=1.0/(1+EXP(-1*X2(J)))
 60       CONTINUE

***** Output activation
          DO 80 K=1,NO
            X3(K)=0.0
```

```
            DO 90 J=1,NM1
              X3(K)=X3(K)+X2(J)*W2(J,K,1)
 90         CONTINUE
*           (Avoid over/under flows)
            IF (X3(K) .GT. 60.0) X3(K)=60.0
            IF (X3(K) .LT. -60.0) X3(K)=-60.0
            X3(K)=1.0/(1+EXP(-1*X3(K)))
            XERR(LL)=XERR(LL)+ABS(D3(MM,K)-X3(K))

***** Determine row of true population for confusion matrix
            IF (D3(MM,K) .EQ. 1.0) ROW=K
 80       CONTINUE

***** Determine maximum output and column of classified
***** population for confusion matrix
          MAX=0.0
          DO 85 K=1,NO
            IF (X3(K) .GT. MAX) THEN
              MAX=X3(K)
              COL=K
            ENDIF
85        CONTINUE
          MAX=0.0
          CONF(LL,ROW,COL)=CONF(LL,ROW,COL) + 1.0

***** Update upper layer weights
          DO 130 J=1,NM+1
            DO 140 K=1,NO
              TERM1=X3(K)*(1-X3(K))*(D3(MM,K)-X3(K))*X2(J)
              TERM2=W2(J,K,2)-W2(J,K,1)
              W2(J,K,3)=W2(J,K,2)+C1*TERM1+C2*TERM2
 140        CONTINUE
            DO 145 K=1,NO
              W2(J,K,1)=W2(J,K,2)
              W2(J,K,2)=W2(J,K,3)
 145        CONTINUE
 130      CONTINUE

***** Update lower layer weights
          DO 147 J=1,NM
            DO 150 I=1,NI+1
              TEMP=0.0
              DO 160 K=1,NO
                TEMP=TEMP+(D3(MM,K)-X3(K))*X3(K)*(1-X3(K))*W2(J,K,2)
 160          CONTINUE
              TERM1=X2(J)*(1-X2(J))*X1(MM,I)*TEMP
              TERM2=W1(J,K,2)-W1(J,K,1)
              W1(I,J,3)=W1(I,J,2)+C1*TERM1+C2*TERM2
 150        CONTINUE
            DO 1 55 I=1,NI+1
      W1(I,J,1)=W1(I,J,2)
```

```
            W1(I,J,2)=W1(I,J,3)
 155       CONTINUE
 147     CONTINUE
 50     CONTINUE

***** Additional data is run through the neural network to observe
***** behavior of the neural network on test data.
        DO 51 MM=1,NXTS
          Y2(NM1)=1.0
          DO 61 J=1,NM
            Y2(J)=0.0
            DO 71 I=1,NI1
              Y2(J)=Y2(J)+Y1(MM,I)*W1(I,J,2)
 71         CONTINUE
*           (Avoid over/under flows)
            IF (Y2(J) .GT. 60.0) Y2(J)=60.0
            IF (Y2(J) .LT. -60.0) Y2(J)=-60.0
            Y2(J)=1.0/(1+EXP(-1*Y2(J)))
 61       CONTINUE

          DO 81 K=1,NO
            Y3(K)=0.0
            DO 91 J=1,NM1
              Y3(K)=Y3(K)+Y2(J)*W2(J,K,1)
 91         CONTINUE
*           (Avoid over/under flows)
            IF (Y3(K) .GT. 60.0) Y3(K)=60.0
            IF (Y3(K) .LT. -60.0) Y3(K)=-60.0
            Y3(K)=1.0/(1+EXP(-1*Y3(K)))
            YERR(LL)=YERR(LL)+ABS(E3(MM,K)-Y3(K))

***** Determines row of true population for test confusion matrix
            IF (E3(MM,K) .EQ. 1.0) RW2=K
 81       CONTINUE

***** Determines maximum output and column of classified
***** population for confusion matrix
          MAX=0.0
          DO 88 K=1,NO
            IF (Y3(K) .GT. MAX) then
              MAX=Y3(K)
              CL2=K
            ENDIF
 88         CONTINUE
          CCONFT(LL,RW2,CL2)=CCONFT(LL,RW2,CL2) + 1.0
 51     CONTINUE

***** Send error history and probability of misclassification to a file
        DO 171 I=1,NO
          TEMP=0.0
          TEMPT=0.0
```

```
          DO 175 J=1,NO
            TEMP=TEMP+CONF(LL,I,J)
            TEMPT=TEMPT+CCONFT(LL,I,J)
 175       CONTINUE
          DO 172 J=1,NO
            CONF(LL,I,J)=CONF(LL,I,J)*100.0/TEMP
            CCONFT(LL,I,J)=CCONFT(LL,I,J)*100.0/TEMPT
 172       CONTINUE
 171      CONTINUE
         XERR(LL)=XERR(LL)/REAL(NXTR)
         YERR(LL)=YERR(LL)/REAL(NXTS)
         WRITE(19,700) LL, XERR(LL),YERR(LL)
         WRITE(*,*)LL,XERR(LL),YERR(LL)
         WRITE(20,706) LL,((CONF(LL,I,J),J=1,NO), I=1,NO),
     ;                  LL,((CCONFT(LL,I,J),J=1,NO), I=1,NO)
         WRITE(20,*)
         MISS(LL)=REAL(NO)*100.0
         MISST(LL)=REAL(NO)*100.0
         DO 196 I=1,NO
           MISS(LL)=MISS(LL)-CONF(LL,I,I)
           MISST(LL)=MISST(LL)-CCONFT(LL,I,I)
 196     CONTINUE
           MISS(LL)=MISS(LL)/REAL(NO)
           MISST(LL)=MISST(LL)/REAL(NO)
           WRITE(24,*) LL,MISS(LL),MISST(LL)
           WRITE(*,*)LL,MISS(LL),MISST(LL)
 11    CONTINUE

***** END OF EPOCH

       NEF=NE

***** Save weights to a file
*      Lower layer weights from input to middle layer
       WRITE(21,*) 'LOWER WEIGHTS: BETWEEN INPUT LAYER AND HIDDEN LAYER'
       DO 180 I=1,NI+1
          WRITE(21,710) (W1(I,J,3),J=1,NM)
 180   CONTINUE
*      Upper layer weights from middle layer to output layer
       WRITE(21,*) 'UPPER WEIGHTS: BETWEEN HIDDEN LAYER AND OUTPUT LAYER'
       DO 190 I=1,NM+1
         WRITE(21,720) (W2(I,J,3),J=1,NO)
 190    CONTINUE

 700    FORMAT(I4,1X,F12.6,1X,F12.6)

 706    FORMAT('itr=',I4,' train:',2(F6.2,1X)/'              ',2(F6.2,1X)
     ;/'itr=',I4,'  test:',2(F6.2,1X)/'              ',2(F6.2,1X))
 710    FORMAT(10(1X,F12.6))
 715    FORMAT(I4,1X,F6.2,1X,F6.2)
 720    FORMAT(10(1X,F12.6))
```

```
      RETURN
      END



*1111111111111111111111111111111111111111111111111111111111111111111111111111111
*
*  Subroutine SALIENCY:  This subroutine computes the saliency of each
*                        of the features based on the weights of the final
*                        calculated net.  Saliency is a measure of the
*                        significance that a feature has on the output of
*                        a multilayer perceptron.
*
*     XSAL(I,J):  Array of input data transformed to saliency
*                 data configuration
*     X2SAL(J):  Activation of hidden layer on saliency data
*     X3SAL(K):  Activation of output layer on saliency data
*     XDIV(J,K):  Division of input feature j into k subdivisions
*     SAL(J):  Saliency of input feature j
*     NDIV:  Number of divisions for range of input feature
*     NDIV1:  Number of divisions plus one for bias
*
*1111111111111111111111111111111111111111111111111111111111111111111111111111111

      SUBROUTINE SALIENCY(WT1,WT2,X1,IREPTR,IREPTS,
     ;NVAR1,NGRP2,NDIV,
     ;NDIV1,NSAL,NM1,NM,X2SAL,X3SAL,XSAL,XDIV,SAL)

      INTEGER NGRP2,IREPTR,IREPTS,NM1,NM,NGRP2,NVAR1

      REAL X1(IREPTR,NVAR1),WT1(NVAR1,NM,3),
     ;WT2(NM1,NGRP2,3),
     ;XDIV(NVAR1,NDIV1),
     ;XSAL(NSAL,NVAR1),X2SAL(NM1),X3SAL(NGRP2),
     ;SUM,DERIV,SAL(NVAR1)
     ;,MAX,MIN

***** Creating array of divisions over range of each factor
      DO 100 JJ = 1,NVAR1
        DO 110 II=1,IREPTR
          MAX=0.0
          MIN=1000.0
          IF ((X1(II,JJ).NE. -1.0) .AND. (X1(II,JJ).NE.1.0)) GO TO 120
 110    CONTINUE
        GO TO 130

 120    DO 140 I2=1,IREPTR
          IF (X1(I2,JJ).GT. MAX) MAX = X1(I2,JJ)
          IF (X1(I2,JJ).LT. MIN) MIN = X1(I2,JJ)
 140    CONTINUE
```

```
          DO 160 K = 0,NDIV
            XDIV(JJ,K+1) = MIN+((MAX-MIN)/NDIV)*K
 160      CONTINUE

        GO TO 100

 130      HDIV = NDIV/2
          DO 190 K = 0,HDIV
            XDIV(JJ,K+1) = -1.0
 190      CONTINUE
          DO 200 K = (HDIV+1),NDIV
           XDIV(JJ,K+1) = 1.0
 200      CONTINUE
 100  CONTINUE

***** Saliency calculations
 170  DO 210 J = 1,NVAR1

*****  Create saliency array
        SAL(J)=0.0
        NSAL=NDIV1*IREPTR
        DO 220 JJ=1,NVAR1
          DO 230 I=0,(IREPTR-1)
            DO 240 K=0,NDIV
                INT=((NDIV+1)*I)+(K+1)
                IF (J.EQ.JJ) XSAL(INT,JJ) = XDIV(JJ,K+1)
                IF (J.NE.JJ) XSAL(INT,JJ) = X1((I+1),JJ)
 240        CONTINUE
 230      CONTINUE
 220    CONTINUE

***** Calculate activations with fixed weights
        DO 250 MM=1,NSAL
          X2SAL(NM1)=1.0
          DO 260 JJ = 1,NM
            X2SAL(JJ)=0.0
            DO 270 I = 1,NVAR1
              X2SAL(JJ)=X2SAL(JJ)+XSAL(MM,I)*WT1(I,JJ,3)
 270        CONTINUE
            X2SAL(JJ) = 1.0/(1+EXP(-1*X2SAL(JJ)))
 260      CONTINUE
          DO  280 K = 1,NGRP2
            X3SAL(K)=0.0
            DO 290 JJ = 1,NM1
              X3SAL(K) = X3SAL(K)+X2SAL(JJ)*WT2(JJ,K,3)
 290        CONTINUE
            X3SAL(K)=1.0/(1+EXP(-1*X3SAL(K)))
 280      CONTINUE
          DERIV=0.0
          SUM2=0.0
```

```
***** Saliency calculation over all vectors, divisions, and outputs
        DO 300 I = 1,NGRP2
           SUM=0.0
           DO 310 M2 = 1,NM1
             SUM = SUM+(WT2(M2,I,3)*X2SAL(M2)
    ;*(1-X2SAL(M2))*WT1(J,M2,3))
 310        CONTINUE
            SUM2 = ABS(SUM * X3SAL(I)*(1-X3SAL(I)))
            DERIV = DERIV + SUM2
 300      CONTINUE
          SAL(J) = SAL(J)+ DERIV

 250    CONTINUE
 210  CONTINUE

***** Write saliency to a file
      DO 320 J = 1,NVAR1
        WRITE(22,730) J,SAL(J)
 320  CONTINUE

 730   FORMAT('input=',I4,'saliency=',F10.2)

      RETURN
      END


*1111111111111111111111111111111111111111111111111111111111111111111111111111
*
*  Subroutine VALID:  This subroutine takes unclassified data, runs it
*                     through the net and reports the classification
*                     of each vector
*
*     G3:  Vector of Desired Outputs for validation set
*     V1:  Vector of Input features for validation set
*     V2:  Vector of Outputs of Hidden Layer after through sigmoid for training
*     V3:  Vector of Outputs of Output Layer after through sigmoid for training
*     W1(i,j,l):  Weight connecting input layer i to hidden layer j
*                 l=3 means weight corresponds to current iteration
*                 l=2 means weight corresponds to last iteration
*                 l=1 means weight corresponds to iteration before last
*     W2(j,k,l):  Weight connecting hidden layer j to output layer k
*                 l=3 means weight corresponds to current iteration
*                 l=2 means weight corresponds to last iteration
*                 l=1 means weight corresponds to iteration before last
*     NVAR:  Counter for number of input nodes
*     NM:  Counter for number of hidden nodes
*     NGRP2:  Counter for number of output nodes
*     IREPVL:  Number of vectors in validation set
*     VERR:  Validation set error for each epoch
*     VCONFT:  Confusion matrix for validation set
*
```

```
*1111111111111111111111111111111111111111111111111111111111111111111111111111

       SUBROUTINE VALID(V1,V2,V3,VAL,W1,W2,IREPVL,NVAR
     ;,NVAR1,NGRP2,NM,NM1,NE,RANGE,MAX,MIN,G3,VCONFT)

       INTEGER NVAR,NVAR1,IREPVL,NGRP2,NE

       REAL VAL(IREPVL,NVAR1),V1(IREPVL,NVAR1),
     ;W1(NVAR1,NM,3),W2(NM1,NGRP2,3),V2(NM1),V3(NGRP2),
     ;RANGE(NVAR),MAX(NVAR),MIN(NVAR),MAX2,G3(IREPVL,NGRP2),
     ;VCONFT(NGRP2,NGRP2)

***** Read in validation set
       DO 10 I=1,IREPVL
          READ(16,100)(VAL(I,J), J=1,NVAR1)
          WRITE(*,*)'READ VALID VECTOR',I
  10   CONTINUE

***** It is necessary to change the following format statement
***** to read a specific validation set.
 100    FORMAT(F7.4,2X,F7.2,2X,F7.3,2X,F7.5,2X,F7.5,2X,F7.5)

***** Write feature vectors and desired outputs to temporary files
       DO 70 I7=1,IREPVL
         DO 80 J3= 1,NVAR
           V1(I7,J3)=VAL(I7,J3)
 80     CONTINUE
          V1(I7,NVAR1)=1.0
 70    CONTINUE

       DO 55 J = 1,NVAR
         DO 30 I3=1,IREPVL
           V1(I3,J)=(V1(I3,J)-MIN(J))/RANGE(J)
 30      CONTINUE
 55    CONTINUE

       DO 400 I=1,IREPVL
         IF (VAL(I,NVAR1).EQ.1.0) G3(I,1)=1.0
         IF (VAL(I,NVAR1).EQ.1.0) G3(I,2)=0.0
         IF (VAL(I,NVAR1).EQ.0.0) G3(I,2)=1.0
         IF (VAL(I,NVAR1).EQ.0.0) G3(I,1)=0.0
 400   CONTINUE

***** Initialize confusion matrix
         DO 200 I=1,NGRP2
           DO 210 J=1,NGRP2
             VCONFT(I,J)=0.0
 210       CONTINUE
 200     CONTINUE
```

```
***** Calculate activations for hidden layer
        VERR=0.0
        DO 51 MM=1,IREPVL
          V2(NM1)=1.0
          DO 61 J=1,NM
            V2(J)=0.0
            DO 71 I=1,NVAR1
              V2(J)=V2(J)+V1(MM,I)*W1(I,J,2)
 71         CONTINUE
*           (Avoid over/under flows)
            IF (V2(J) .GT. 60.0) V2(J)=60.0
            IF (V2(J) .LT. -60.0) V2(J)=-60.0
            V2(J)=1.0/(1+EXP(-1*V2(J)))

 61       CONTINUE

***** Calculate activations for output layer
          DO 81 K=1,NGRP2
            V3(K)=0.0
            DO 91 J=1,NM1
              V3(K)=V3(K)+V2(J)*W2(J,K,1)
 91         CONTINUE
*           (Avoid over/under flows)
            IF (V3(K) .GT. 60.0) V3(K)=60.0
            IF (V3(K) .LT. -60.0) V3(K)=-60.0
            V3(K)=1.0/(1+EXP(-1*V3(K)))
            VERR=VERR+ABS(G3(MM,K)-V3(K))
            IF (G3(MM,K) .EQ. 1.0) RW2=K
 81       CONTINUE

***** Determines maximum output
          MAX2=0.0
          DO 88 K=1,NGRP2
            IF (V3(K) .GT. MAX2) THEN
              MAX2=V3(K)
              CL2=K
            ENDIF
 88         CONTINUE
          VCONFT(RW2,CL2)=VCONFT(RW2,CL2)+1.0
 51     CONTINUE

***** Construc confusion matrix
        DO 401 I=1,NGRP2
          VTEMP=0.0
          DO 402 J=1,NGRP2
            VTEMP=VTEMP+VCONFT(I,J)
 402      CONTINUE
          DO 403 J=1,NGRP2
            VCONFT(I,J)=VCONFT(I,J)*100.0/VTEMP
 403      CONTINUE
 401    CONTINUE
```

```
***** Calculate average error per validation vector
        VERR=VERR/IREPVL

***** Write error and confusion matrix to a file
        WRITE(17,*)'VALIDATION ERROR',VERR
        WRITE(17,*)((VCONFT(I,J),J=1,NGRP2), I=1,NGRP2)

        RETURN
        END


*1111111111111111111111111111111111111111111111111111111111111111111111111111
*
*  Subroutine CORRELATE:  This subroutine calculates the second-order
*                         correlation matrix of the product of two
*                         inputs with the output.  The resultant matrices
*                         are helpful in determining which second-order
*                         terms should be included as input.
*
*     X1:  Vector of Input features for training set
*     U1:  Vector of Input features for training set
*     U2:  Vector of Outputs of Hidden Layer after through sigmoid for training
*     U3:  Vector of Outputs of Output Layer after through sigmoid for training
*     W1(i,j,l):  Weight connecting input layer i to hidden layer j
*                 l=3 means weight corresponds to current iteration
*                 l=2 means weight corresponds to last iteration
*                 l=1 means weight corresponds to iteration before last
*     W2(j,k,l):  Weight connecting hidden layer j to output layer k
*                 l=3 means weight corresponds to current iteration
*                 l=2 means weight corresponds to last iteration
*                 l=1 means weight corresponds to iteration before last
*     NVAR:  Counter for number of input nodes
*     NM:  Counter for number of hidden nodes
*     NGRP2:  Counter for number of output nodes
*     IREPTR:  Number of vectors in training set
*
*1111111111111111111111111111111111111111111111111111111111111111111111111111

      SUBROUTINE CORRELATE(X1,W1,W2,IREPTR,NVAR1,NM1,NM,NGRP2,
     ;U2,U3,NVAR,NM,XX,YY,YSUM,YBAR,XSUM,XBAR,COR2)

      INTEGER NVAR,NVAR1,IREPTR,NGRP2,NM

      REAL U2(NM1),X1(IREPTR,NVAR1),W1(NVAR1,NM,3),W2(NM1,NGRP2,3),
     ;U3(NGRP2),XX(IREPTR,NVAR1),YY(IREPTR,NGRP2),YSUM(NGRP2),
     ;YBAR(NGRP2),XSUM(NVAR1),XBAR(NVAR1),COR2(NGRP2,NVAR1,NVAR1)

***** Initialize summing arrays
        DO 300 I=1,NGRP2
          YSUM(I)=0.0
```

```
 300    CONTINUE
        DO 310 I=1,NVAR
          XSUM(I)=0.0
 310    CONTINUE

***** Calculate activations for hidden layer
        DO 51 MM=1,IREPTR
          U2(NM1)=1.0
          DO 61 J=1,NM
            U2(J)=0.0
            DO 71 I=1,NVAR1
              U2(J)=U2(J)+X1(MM,I)*W1(I,J,2)
 71         CONTINUE
*           (Avoid over/under flows)
            IF (U2(J) .GT. 60.0) U2(J)=60.0
            IF (U2(J) .LT. -60.0) U2(J)=-60.0
            U2(J)=1.0/(1+EXP(-1*U2(J)))
 61       CONTINUE

***** Calculate activations for output layer
          DO 81 K=1,NGRP2
            U3(K)=0.0
            DO 91 J=1,NM1
              U3(K)=U3(K)+U2(J)*W2(J,K,1)
 91         CONTINUE
*           (Avoid over/under flows)
            IF (U3(K) .GT. 60.0) U3(K)=60.0
            IF (U3(K) .LT. -60.0) U3(K)=-60.0
            U3(K)=1.0/(1+EXP(-1*U3(K)))
            YY(MM,K)=U3(K)
 81       CONTINUE
 51   CONTINUE

***** Read training vectors into xx array
      DO 200 I=1,IREPTR
        DO 210 J=1,NVAR1
          XX(I,J)=X1(I,J)
 210    CONTINUE
 200  CONTINUE

***** Calculate the mean of the outputs for output nodes
      DO 100 I=1,NGRP2
        DO 110 S=1,IREPTR
          YSUM(I)=YSUM(I)+YY(S,I)
 110    CONTINUE
        YBAR(I)=YSUM(I)/IREPTR
 100  CONTINUE

***** Calculate the mean of the input features
      DO 120 I=1,NVAR
        DO 130 S=1,IREPTR
```

```
          XSUM(I)=XSUM(I)+XX(S,I)
 130    CONTINUE
        XBAR(I)=XSUM(I)/IREPTR
 120  CONTINUE

***** Calculate the correlation of second order inputs to outputs
      DO 10 I=1,NGRP2
        DO 20 J=1,NVAR
          DO 30 K=1,NVAR
            DO 40 S=1,IREPTR
              COR2(I,J,K)=COR2(I,J,K)+(YY(S,I)-YBAR(I))*
     ;(XX(S,J)-XBAR(J))*(XX(S,K)-XBAR(K))
 40         CONTINUE
 30       CONTINUE
 20     CONTINUE
 10   CONTINUE

***** Write correlations to a file
      DO 400 J=1,NVAR
        DO 410 K=1,NVAR
          WRITE(18,*)'CORRELATIONS FOR TERMS',J,' *',K
          WRITE(18,*)((COR2(I,J,K)/IREPTR),I=1,NGRP2)
 410    CONTINUE
 400  CONTINUE


      RETURN
      END

***** END OF PROGRAM*****
```

Appendix C. *Pilot Data Configuration Program*

```
************************************************************************
*
*  PILOT DATA CONFIGURATION PROGRAM
*  Capt Lisa M. Belue
*
*  Date Last Modified:  29 February 1992
*
*  Program: datfig2.f
*
*  Purpose:  The purpose of this program is to read in data provided
*            by AF/DPXA on the attributes of individual pilots and format
*            it for use by both a multilayer perceptron classifier and
*            a discriminant analysis classifier.
*
************************************************************************
      EXTERNAL RNUNF
      INTRINSIC REAL,NINT

      INTEGER NR1,NR2,NCOL,N1COL,N1TRAIN,N1TEST,N1VAL,N2TRAIN,
     ;N2TEST,N2VAL

      PARAMETER(NCOL=23,N1COL=26,NR1=10162,NR2=9425,
     ;N1TRAIN=1000,N1TEST=1000,N1VAL=8162,N2TRAIN=1000,
     ;N2TEST=1000,N2VAL=7425)

      INTEGER CHOICE(NR1),CHOICE2(NR2)

      REAL NOISE

      CHARACTER*15 DIGIT,SECON,FIRST,DAFSC,PAFSC

      CHARACTER*20 C88(NR1,NCOL)
      CHARACTER*20 C89(NR2,NCOL)
      CHARACTER*20 X88(NR1,N1COL)
      CHARACTER*20 X89(NR2,N1COL)
      CHARACTER*20 XTR88(N1TRAIN,N1COL)
      CHARACTER*20 XTS88(N1TEST,N1COL)
      CHARACTER*20 XVL88(N1VAL,N1COL)
      CHARACTER*20 XTR89(N2TRAIN,N1COL)
      CHARACTER*20 XTS89(N2TEST,N1COL)
      CHARACTER*20 XVL89(N2VAL,N1COL)

      OPEN(UNIT=11,FILE='pilot88.dat',STATUS='UNKNOWN')
      OPEN(UNIT=12,FILE='pilot89.dat',STATUS='UNKNOWN')
      OPEN(UNIT=13,FILE='train88.dat',STATUS='UNKNOWN')
      OPEN(UNIT=14,FILE='test88.dat',STATUS='UNKNOWN')
```

```
      OPEN(UNIT=15,FILE='val88.dat',STATUS='UNKNOWN')
      OPEN(UNIT=16,FILE='train89.dat',STATUS='UNKNOWN')
      OPEN(UNIT=17,FILE='test89.dat',STATUS='UNKNOWN')
      OPEN(UNIT=18,FILE='val89.dat',STATUS='UNKNOWN')
      OPEN(UNIT=23,FILE='train88.sas',STATUS='UNKNOWN')
      OPEN(UNIT=24,FILE='test88.sas',STATUS='UNKNOWN')
      OPEN(UNIT=25,FILE='val88.sas',STATUS='UNKNOWN')
      OPEN(UNIT=26,FILE='train89.sas',STATUS='UNKNOWN')
      OPEN(UNIT=27,FILE='test89.sas',STATUS='UNKNOWN')
      OPEN(UNIT=28,FILE='val89.sas',STATUS='UNKNOWN')

       DO  10 I=1,NR1
        READ(11,100)(C88(I,J), J=1,NCOL)
 10    CONTINUE

      DO 20 I=1,NR2
        READ(12,100)(C89(I,J), J=1,NCOL)
 20    CONTINUE

*************TAFMSD,ADSCDA,MARSTAT,DEPN,RDTM,RPI,GRADEA,RETPROG,
******PME,MAJCOM,DOB,PASCBPO,ACADLVL,ACADSPEC,DAFSC
******PAFSC,PRIORSV,SOC,RACE,COMP,SEX,FLYMONTH,RETAIN

  100  FORMAT(A4,A4,A1,A2,A1,A1,A2,A1,
     ;A1,A2,A4,A2,A1,A2,2X,A6,
     ;A6,A1,A1,A1,A1,A1,A3,A1)


*******************************************************************************
***************************** FY 88 DATA ***************************************
*******************************************************************************

      DO 30 I=1,NR1

      WRITE(*,*)'FY88 DATA      ITERATION = ',I

**********************************
********** TAFMSD ****************
**********************************

        X88(I,1)=C88(I,1)


************************************************
********** ADSCDA ********************************
************************************************

        X88(I,2)=C88(I,2)

*****************************
********** MARSTAT **********
```

```
*****************************

        IF (C88(I,3).EQ.'M') THEN
          X88(I,3)='-1 -1'
        ELSE IF (C88(I,3).EQ.'S') THEN
          X88(I,3)=' 1 -1'
        ELSE IF (C88(I,3).EQ.'D') THEN
          X88(I,3)='-1  1'
        ELSE
          X88(I,3)=' 1  1'
        ENDIF

*************************
********** DEPN **********
*************************

        X88(I,4)=C88(I,4)
        IF (X88(I,4).EQ.' .') X88(I,4)=' 0'

*************************
********** RDTM **********
*************************

        IF (C88(I,5).EQ.'A') THEN
          X88(I,5)=' 1  1  1'
        ELSE IF (C88(I,5).EQ.'D') THEN
          X88(I,5)='-1  1  1'
        ELSE IF (C88(I,5).EQ.'E') THEN
          X88(I,5)=' 1 -1  1'
        ELSE IF (C88(I,5).EQ.'F') THEN
          X88(I,5)='-1 -1  1'
        ELSE IF (C88(I,5).EQ.'G') THEN
          X88(I,5)=' 1  1 -1'
        ELSE IF (C88(I,5).EQ.'H') THEN
          X88(I,5)='-1  1 -1'
        ELSE IF (C88(I,5).EQ.'J') THEN
          X88(I,5)=' 1 -1 -1'
        ELSE IF (C88(I,5).EQ.'K') THEN
          X88(I,5)='-1 -1 -1'
        ELSE IF (C88(I,5).EQ.'L') THEN
          X88(I,5)='-1 -1 -1'
        ELSE
          X88(I,5)='-1 -1 -1'
        ENDIF

*************************
********** RPI **********
*************************

        IF (C88(I,6).EQ.'0') X88(I,6)=' 1  1  1'
        IF (C88(I,6).EQ.'1') X88(I,6)='-1  1  1'
```

```
        IF (C88(I,6).EQ.'2') X88(I,6)='-1 -1 -1'
        IF (C88(I,6).EQ.'3') X88(I,6)=' 1 -1  1'
        IF (C88(I,6).EQ.'4') X88(I,6)='-1 -1  1'
        IF (C88(I,6).EQ.'5') X88(I,6)='-1 -1 -1'
        IF (C88(I,6).EQ.'6') X88(I,6)=' 1  1 -1'
        IF (C88(I,6).EQ.'7') X88(I,6)='-1  1 -1'
        IF (C88(I,6).EQ.'8') X88(I,6)=' 1 -1 -1'
        IF (C88(I,6).EQ.'9') X88(I,6)='-1 -1 -1'

*****************************
********** GRADEA ***********
*****************************

        X88(I,7)=C88(I,7)

*****************************
********** RETPROG **********
*****************************

        IF (C88(I,8).EQ.'1') THEN
          X88(I,8)='-1 -1'
        ELSE IF (C88(I,8).EQ.'2') THEN
          X88(I,8)='-1  1'
        ELSE IF (C88(I,8).EQ.'3') THEN
          X88(I,8)=' 1 -1'
        ELSE
          X88(I,8)=' 1  1'
        ENDIF

*************************
********** PME **********
*************************

        IF (C88(I,9).EQ.'A') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'B') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'C') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'D') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'E') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'F') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'G') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'H') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'J') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'K') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'L') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'M') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'N') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'P') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'Q') X88(I,9)='-1 -1'
        IF (C88(I,9).EQ.'R') X88(I,9)='-1 -1'
        IF (C88(I,9).EQ.'S') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'T') X88(I,9)=' 1  1'
```

```
        IF (C88(I,6).EQ.'2') X88(I,6)='-1 -1 -1'
        IF (C88(I,6).EQ.'3') X88(I,6)=' 1 -1  1'
        IF (C88(I,6).EQ.'4') X88(I,6)='-1 -1  1'
        IF (C88(I,6).EQ.'5') X88(I,6)='-1 -1 -1'
        IF (C88(I,6).EQ.'6') X88(I,6)=' 1  1 -1'
        IF (C88(I,6).EQ.'7') X88(I,6)='-1  1 -1'
        IF (C88(I,6).EQ.'8') X88(I,6)=' 1 -1 -1'
        IF (C88(I,6).EQ.'9') X88(I,6)='-1 -1 -1'

*****************************
********** GRADEA ***********
*****************************

        X88(I,7)=C88(I,7)

*****************************
********** RETPROG **********
*****************************

        IF (C88(I,8).EQ.'1') THEN
          X88(I,8)='-1 -1'
        ELSE IF (C88(I,8).EQ.'2') THEN
          X88(I,8)='-1  1'
        ELSE IF (C88(I,8).EQ.'3') THEN
          X88(I,8)=' 1 -1'
        ELSE
          X88(I,8)=' 1  1'
        ENDIF

************************
********** PME **********
************************

        IF (C88(I,9).EQ.'A') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'B') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'C') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'D') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'E') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'F') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'G') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'H') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'J') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'K') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'L') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'M') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'N') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'P') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'Q') X88(I,9)='-1 -1'
        IF (C88(I,9).EQ.'R') X88(I,9)='-1 -1'
        IF (C88(I,9).EQ.'S') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'T') X88(I,9)=' 1  1'
```

```
        IF (C88(I,9).EQ.'U') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'V') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'W') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'X') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'Z') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'1') X88(I,9)='-1 -1'
        IF (C88(I,9).EQ.'2') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'3') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'4') X88(I,9)=' 1 -1'
        IF (C88(I,9).EQ.'5') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'6') X88(I,9)='-1  1'
        IF (C88(I,9).EQ.'7') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'8') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.'9') X88(I,9)=' 1  1'
        IF (C88(I,9).EQ.' ') X88(I,9)=' 1  1'

**********************************
********** DESIRED PME **********
**********************************
********* COMPLETED = 1 *********
******* NOT COMPLETED =-1 *******
**********************************

      IF (((X88(I,9).EQ.'-1 -1').OR.(X88(I,9).EQ.'-1  1').OR.
     ;(X88(I,9).EQ.' 1 -1')).AND.(X88(I,1).LT.8112)) X88(I,10)=' 1'

      IF (((X88(I,9).EQ.'-1  1').OR.
     ;(X88(I,9).EQ.' 1 -1')).AND.(X88(I,1).LT.7512)) X88(I,10)=' 1'

      IF (X88(I,9).EQ.' 1 -1') X88(I,10)=' 1'

      IF (((X88(I,9).NE.'-1 -1').AND.(X88(I,9).NE.'-1  1').AND.
     ;(X88(I,9).NE.' 1 -1')).AND.(X88(I,1).LT.8112)) X88(I,10)='-1'

      IF (((X88(I,9).NE.'-1  1').AND.
     ;(X88(I,9).NE.' 1 -1')).AND.(X88(I,1).LT.7512)) X88(I,10)='-1'

      IF ((X88(I,9).NE.' 1 -1').AND.(X88(I,1).LT.7112)) X88(I,10)='-1'


****************************
********** MAJCOM **********
****************************

        IF (C88(I,10).EQ.'0A') X88(I,11)=' 1  1  1  1'
        IF (C88(I,10).EQ.'0B') X88(I,11)='-1 -1 -1  1'
        IF (C88(I,10).EQ.'0C') X88(I,11)='-1 -1 -1 -1'
        IF (C88(I,10).EQ.'0D') X88(I,11)='-1  1  1  1'
        IF (C88(I,10).EQ.'0E') X88(I,11)='-1 -1 -1 -1'
        IF (C88(I,10).EQ.'0F') X88(I,11)=' 1 -1  1  1'
        IF (C88(I,10).EQ.'0G') X88(I,11)='-1 -1 -1 -1'
```

```
IF (C88(I,10).EQ.'0H') X88(I,11)='-1 -1  1  1'
IF (C88(I,10).EQ.'0I') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'0J') X88(I,11)=' 1  1 -1  1'
IF (C88(I,10).EQ.'0K') X88(I,11)='-1  1 -1  1'
IF (C88(I,10).EQ.'0L') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'0M') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'0N') X88(I,11)=' 1 -1 -1  1'
IF (C88(I,10).EQ.'0O') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'0P') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'0Q') X88(I,11)='-1  1 -1 -1'
IF (C88(I,10).EQ.'0R') X88(I,11)=' 1  1  1 -1'
IF (C88(I,10).EQ.'0S') X88(I,11)='-1  1  1 -1'
IF (C88(I,10).EQ.'0T') X88(I,11)=' 1 -1  1 -1'
IF (C88(I,10).EQ.'0U') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'0V') X88(I,11)=' 1 -1 -1 -1'
IF (C88(I,10).EQ.'0X') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'0Y') X88(I,11)='-1 -1  1 -1'
IF (C88(I,10).EQ.'0Z') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'01') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'02') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'03') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'05') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'06') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'07') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'08') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'09') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'1S') X88(I,11)=' 1  1 -1 -1'
IF (C88(I,10).EQ.'1W') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'1X') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'2A') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2C') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'2E') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2F') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2G') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2H') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2I') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'2J') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'2K') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2L') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2M') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2N') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2P') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2R') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'2W') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'3C') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'3F') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'3G') X88(I,11)='-1  1  1  1'
IF (C88(I,10).EQ.'3H') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'3I') X88(I,11)='-1 -1 -1 -1'
IF (C88(I,10).EQ.'3R') X88(I,11)='-1 -1 -1  1'
IF (C88(I,10).EQ.'3S') X88(I,11)='-1 -1 -1  1'
```

```
        IF (C88(I,10).EQ.'3V') X88(I,11)='-1 -1 -1  1'
        IF (C88(I,10).EQ.'34') X88(I,11)='-1 -1 -1 -1'
        IF (C88(I,10).EQ.'77') X88(I,11)='-1 -1 -1  1'
        IF (C88(I,10).EQ.'8J') X88(I,11)='-1 -1 -1 -1'
        IF (C88(I,10).EQ.'88') X88(I,11)='-1 -1 -1  1'
        IF (C88(I,10).EQ.'YY') X88(I,11)='-1 -1 -1 -1'

*****************************
********** DOB (AGE) ********
*****************************

        X88(I,12)=C88(I,11)

*****************************
********** PASCBPO **********
*****************************

        IF(C88(I,12).EQ.'AF') X88(I,13)=' 1 -1  1  1'
        IF(C88(I,12).EQ.'AH') X88(I,13)=' 1  1  1  1'
        IF(C88(I,12).EQ.'AK') X88(I,13)='-1  1  1 -1'
        IF(C88(I,12).EQ.'AM') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'AT') X88(I,13)='-1  1  1  1'
        IF(C88(I,12).EQ.'AU') X88(I,13)=' 1  1  1 -1'
        IF(C88(I,12).EQ.'AX') X88(I,13)=' 1  1  1  1'
        IF(C88(I,12).EQ.'AY') X88(I,13)=' 1  1  1  1'
        IF(C88(I,12).EQ.'A2') X88(I,13)='-1  1  1 -1'
        IF(C88(I,12).EQ.'A3') X88(I,13)='-1  1  1 -1'
        IF(C88(I,12).EQ.'A4') X88(I,13)=' 1 -1 -1  1'
        IF(C88(I,12).EQ.'A5') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'A6') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'A7') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'A8') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'A9') X88(I,13)=' 1 -1 -1  1'
        IF(C88(I,12).EQ.'BD') X88(I,13)=' 1  1 -1 -1'
        IF(C88(I,12).EQ.'BF') X88(I,13)='-1  1  1 -1'
        IF(C88(I,12).EQ.'BB') X88(I,13)=' 1  1  1  1'
        IF(C88(I,12).EQ.'BH') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'BL') X88(I,13)=' 1  1  1  1'
        IF(C88(I,12).EQ.'BN') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'BP') X88(I,13)=' 1  1  1 -1'
        IF(C88(I,12).EQ.'BV') X88(I,13)='-1 -1  1 -1'
        IF(C88(I,12).EQ.'BX') X88(I,13)=' 1 -1  1 -1'
        IF(C88(I,12).EQ.'B2') X88(I,13)=' 1  1 -1 -1'
        IF(C88(I,12).EQ.'B3') X88(I,13)=' 1  1 -1 -1'
        IF(C88(I,12).EQ.'B4') X88(I,13)=' 1  1 -1 -1'
        IF(C88(I,12).EQ.'B5') X88(I,13)=' 1  1 -1 -1'
        IF(C88(I,12).EQ.'B6') X88(I,13)=' 1  1 -1 -1'
        IF(C88(I,12).EQ.'B7') X88(I,13)=' 1  1  1 -1'
        IF(C88(I,12).EQ.'B8') X88(I,13)=' 1  1  1 -1'
        IF(C88(I,12).EQ.'B9') X88(I,13)=' 1  1  1 -1'
        IF(C88(I,12).EQ.'CC') X88(I,13)=' 1  1  1  1'
```

```
IF(C88(I,12).EQ.'CD') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'CF') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'CH') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'CJ') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'CK') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'CL') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'CO') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'CP') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'CQ') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'CR') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'C2') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'C3') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'C4') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'C5') X88(I,13)='-1  1 -1  1'
IF(C88(I,12).EQ.'C6') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'C7') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'C8') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'C9') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'DF') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'DM') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'DW') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'D2') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'D3') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'D4') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'D5') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'D6') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'D7') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'D8') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'D9') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'EB') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'EC') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'ED') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'EE') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'EH') X88(I,13)=' 1 -1 -1  1'
IF(C88(I,12).EQ.'EJ') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'EL') X88(I,13)=' 1 -1 -1  1'
IF(C88(I,12).EQ.'EM') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'EP') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'E2') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'E3') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'E4') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'E5') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'E7') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'E8') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'E9') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'FB') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'FA') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'FC') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'FE') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'FF') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'FG') X88(I,13)='-1  1 -1  1'
```

```
IF(C88(I,12).EQ.'FH') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'FK') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'FM') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'FN') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'FQ') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'FR') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'FS') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'FT') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'FU') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'FV') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'FW') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'FX') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'F2') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'F3') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'F4') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'F6') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'F7') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'F8') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'GB') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'GC') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'GF') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'GM') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'GW') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'G1') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'G2') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'G3') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'G4') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'G5') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'G6') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'G7') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'G8') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'G9') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'HB') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'HH') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'HL') X88(I,13)='-1  1 -1  1'
IF(C88(I,12).EQ.'HP') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'HS') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'HV') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'H2') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'H3') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'H4') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'H6') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'H7') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'H8') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'H9') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'IC') X88(I,13)=' 1  1 -1  1'
IF(C88(I,12).EQ.'IK') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'IN') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'J2') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'J3') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'J4') X88(I,13)=' 1 -1  1 -1'
```

```
IF(C88(I,12).EQ.'J6') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'J7') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'J8') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'J9') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'KB') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'KF') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'KH') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'KJ') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'KU') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'KV') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'KY') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'K2') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'K3') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'K4') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'K6') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'K7') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'K8') X88(I,13)=' 1 -1  1  1'
IF(C88(I,12).EQ.'K9') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'LA') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'LC') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'LD') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'LE') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'LJ') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'LK') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'LL') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'LP') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'LS') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'LU') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'LW') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'LY') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'L2') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'L3') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'L4') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'L5') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'L6') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'L7') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'L8') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'L9') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'MA') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'MB') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'MD') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'ME') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'MG') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'MH') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'MK') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'ML') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'MN') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'MO') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'MP') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'MT') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'MU') X88(I,13)=' 1  1 -1 -1'
```

```
IF(C88(I,12).EQ.'MW') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'MY') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'M2') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'M3') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'M4') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'M5') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'M6') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'M7') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'M8') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'M9') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'NJ') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'NV') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'N2') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'N3') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'OD') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'OP') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'PD') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'PE') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'PF') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'PJ') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'PS') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'PV') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'RF') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'RJ') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'RM') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'RP') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'RX') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'R1') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'R2') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'R3') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'R4') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'R5') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'R6') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'R7') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'R8') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'R9') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'SB') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'SF') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'SJ') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'SM') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'SP') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'SQ') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'ST') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'S1') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'S2') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'S3') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'S4') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'S5') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'TA') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'TB') X88(I,13)=' 1 -1 -1  1'
IF(C88(I,12).EQ.'TC') X88(I,13)='-1 -1  1 -1'
```

```
IF(C88(I,12).EQ.'TD') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'TE') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'TF') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'TG') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'TH') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'TI') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'TJ') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'TK') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'TL') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'TM') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'TN') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'TO') X88(I,13)='-1  1 -1  1'
IF(C88(I,12).EQ.'TP') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'TQ') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'TR') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'TS') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'TT') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'TV') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'TW') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'TX') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'TY') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'TZ') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'T1') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'T2') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'T3') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'T4') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'T5') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'T6') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'T7') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'T8') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'T9') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'UB') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'UC') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'UD') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'UE') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'UF') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'UG') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'UH') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'UI') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'UJ') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'UK') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'UL') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'UM') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'UN') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'UO') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'UP') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'UQ') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'UR') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'US') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'UT') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'UU') X88(I,13)='-1  1 -1 -1'
```

```
IF(C88(I,12).EQ.'UV') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'UW') X88(I,13)=' 1 -1  1  1'
IF(C88(I,12).EQ.'UX') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'UY') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'UZ') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'U2') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'U3') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'U4') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'U5') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'U6') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'U7') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'U8') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'U9') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'VH') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'VQ') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'WA') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'WC') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'WD') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'WE') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'WF') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'WG') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'WH') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'WI') X88(I,13)=' 1 -1  1  1'
IF(C88(I,12).EQ.'WJ') X88(I,13)='-1  1 -1 -1'
IF(C88(I,12).EQ.'WK') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'WL') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'WM') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'WT') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'WU') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'WV') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'WZ') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'W1') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'W2') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'W3') X88(I,13)='-1 -1 -1  1'
IF(C88(I,12).EQ.'W4') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'W5') X88(I,13)=' 1  1  1 -1'
IF(C88(I,12).EQ.'W6') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'W7') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'W8') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'W9') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'YM') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'YY') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'YZ') X88(I,13)='-1  1  1  1'
IF(C88(I,12).EQ.'ZA') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'ZB') X88(I,13)='-1 -1  1 -1'
IF(C88(I,12).EQ.'ZC') X88(I,13)=' 1 -1  1 -1'
IF(C88(I,12).EQ.'ZE') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'ZG') X88(I,13)=' 1  1  1  1'
IF(C88(I,12).EQ.'ZK') X88(I,13)='-1  1  1 -1'
IF(C88(I,12).EQ.'ZL') X88(I,13)=' 1  1 -1 -1'
IF(C88(I,12).EQ.'ZN') X88(I,13)=' 1  1  1  1'
```

```
      IF(C88(I,12).EQ.'ZS') X88(I,13)='-1 -1  1 -1'
      IF(C88(I,12).EQ.'1A') X88(I,13)=' 1  1  1  1'
      IF(C88(I,12).EQ.'1B') X88(I,13)=' 1 -1 -1  1'
      IF(C88(I,12).EQ.'1C') X88(I,13)='-1  1  1 -1'
      IF(C88(I,12).EQ.'1D') X88(I,13)=' 1 -1  1 -1'
      IF(C88(I,12).EQ.'1F') X88(I,13)='-1  1  1 -1'
      IF(C88(I,12).EQ.'1H') X88(I,13)=' 1  1  1 -1'
      IF(C88(I,12).EQ.'1K') X88(I,13)=' 1  1  1 -1'
      IF(C88(I,12).EQ.'1M') X88(I,13)=' 1  1  1  1'
      IF(C88(I,12).EQ.'1P') X88(I,13)=' 1  1 -1  1'
      IF(C88(I,12).EQ.'1T') X88(I,13)=' 1 -1  1 -1'
      IF(C88(I,12).EQ.'1U') X88(I,13)=' 1  1  1 -1'
      IF(C88(I,12).EQ.'1V') X88(I,13)=' 1  1 -1 -1'
      IF(C88(I,12).EQ.'2B') X88(I,13)=' 1 -1  1 -1'
      IF(C88(I,12).EQ.'2D') X88(I,13)=' 1  1  1 -1'
      IF(C88(I,12).EQ.'7A') X88(I,13)=' 1  1  1 -1'
      IF(C88(I,12).EQ.'7C') X88(I,13)=' 1  1  1  1'
      IF(C88(I,12).EQ.'7S') X88(I,13)='-1 -1 -1  1'
      IF(C88(I,12).EQ.'7V') X88(I,13)='-1 -1  1 -1'
      IF(C88(I,12).EQ.'7Z') X88(I,13)='-1 -1  1 -1'
      IF(C88(I,12).EQ.'8C') X88(I,13)=' 1 -1  1 -1'
      IF(C88(I,12).EQ.'85') X88(I,13)='-1  1  1 -1'
      IF(C88(I,12).EQ.'9C') X88(I,13)='-1  1  1 -1'
      IF(C88(I,12).EQ.'96') X88(I,13)=' 1  1 -1 -1'

*******************************
********** ACAD LVL **********
*******************************

      IF(C88(I,13).EQ.'A') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'B') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'C') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'D') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'E') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'F') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'G') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'H') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'I') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'J') THEN
        X88(I,14)=' 1  1  1'
      ELSE IF(C88(I,13).EQ.'N') THEN
        X88(I,14)=' 1  1  1'
```

```
        ELSE IF(C88(I,13).EQ.'O') THEN
          X88(I,14)='-1  1  1'
        ELSE IF(C88(I,13).EQ.'P') THEN
          X88(I,14)=' 1 -1  1'
        ELSE IF(C88(I,13).EQ.'Q') THEN
          X88(I,14)='-1 -1  1'
        ELSE IF(C88(I,13).EQ.'R') THEN
          X88(I,14)=' 1  1 -1'
        ELSE IF(C88(I,13).EQ.'S') THEN
          X88(I,14)=' 1 -1 -1'
        ELSE IF(C88(I,13).EQ.'T') THEN
          X88(I,14)=' 1 -1 -1'
        ELSE IF(C88(I,13).EQ.'U') THEN
          X88(I,14)=' 1 -1 -1'
        ELSE IF(C88(I,13).EQ.'Y') THEN
          X88(I,14)='-1 -1 -1'
        ELSE
          X88(I,14)='-1 -1 -1'
        ENDIF

*********************************
********** ACAD SPEC **********
*********************************

        IF(C88(I,14).EQ.'0C') X88(I,15)=' 1  1  1  1  1  1  1'
        IF(C88(I,14).EQ.'0G') X88(I,15)='-1  1  1  1  1  1  1'
        IF(C88(I,14).EQ.'0I') X88(I,15)=' 1 -1  1  1  1  1  1'
        IF(C88(I,14).EQ.'0S') X88(I,15)='-1 -1  1  1  1  1  1'
        IF(C88(I,14).EQ.'0Y') X88(I,15)=' 1  1 -1  1  1  1  1'
        IF(C88(I,14).EQ.'1A') X88(I,15)='-1  1 -1  1  1  1  1'
        IF(C88(I,14).EQ.'1B') X88(I,15)=' 1 -1 -1  1  1  1  1'
        IF(C88(I,14).EQ.'1C') X88(I,15)='-1 -1 -1  1  1  1  1'
        IF(C88(I,14).EQ.'1Y') X88(I,15)=' 1  1  1 -1  1  1  1'
        IF(C88(I,14).EQ.'2A') X88(I,15)='-1  1  1 -1  1  1  1'
        IF(C88(I,14).EQ.'2B') X88(I,15)=' 1 -1  1 -1  1  1  1'
        IF(C88(I,14).EQ.'2C') X88(I,15)='-1 -1  1 -1  1  1  1'
        IF(C88(I,14).EQ.'2D') X88(I,15)=' 1  1 -1 -1  1  1  1'
        IF(C88(I,14).EQ.'2E') X88(I,15)='-1  1 -1 -1  1  1  1'
        IF(C88(I,14).EQ.'2F') X88(I,15)=' 1 -1 -1 -1  1  1  1'
        IF(C88(I,14).EQ.'2G') X88(I,15)='-1 -1 -1 -1  1  1  1'
        IF(C88(I,14).EQ.'2H') X88(I,15)=' 1  1  1  1 -1  1  1'
        IF(C88(I,14).EQ.'2I') X88(I,15)='-1  1  1  1 -1  1  1'
        IF(C88(I,14).EQ.'2K') X88(I,15)=' 1 -1  1  1 -1  1  1'
        IF(C88(I,14).EQ.'2Y') X88(I,15)='-1 -1  1  1 -1  1  1'
        IF(C88(I,14).EQ.'3A') X88(I,15)=' 1  1 -1  1 -1  1  1'
        IF(C88(I,14).EQ.'3B') X88(I,15)='-1  1 -1  1 -1  1  1'
        IF(C88(I,14).EQ.'3Y') X88(I,15)=' 1 -1 -1  1 -1  1  1'
        IF(C88(I,14).EQ.'4A') X88(I,15)='-1 -1 -1  1 -1  1  1'
        IF(C88(I,14).EQ.'4B') X88(I,15)=' 1  1  1 -1 -1  1  1'
        IF(C88(I,14).EQ.'4C') X88(I,15)='-1  1  1 -1 -1  1  1'
        IF(C88(I,14).EQ.'4D') X88(I,15)=' 1 -1  1 -1 -1  1  1'
```

```
IF(C88(I,14).EQ.'4E') X88(I,15)='-1 -1  1 -1 -1  1  1'
IF(C88(I,14).EQ.'4F') X88(I,15)=' 1  1 -1 -1 -1  1  1'
IF(C88(I,14).EQ.'4G') X88(I,15)='-1  1 -1 -1 -1  1  1'
IF(C88(I,14).EQ.'4H') X88(I,15)=' 1 -1 -1 -1 -1  1  1'
IF(C88(I,14).EQ.'4I') X88(I,15)='-1 -1 -1 -1 -1  1  1'
IF(C88(I,14).EQ.'4J') X88(I,15)=' 1  1  1  1  1 -1  1'
IF(C88(I,14).EQ.'4K') X88(I,15)='-1  1  1  1  1 -1  1'
IF(C88(I,14).EQ.'4L') X88(I,15)=' 1 -1  1  1  1 -1  1'
IF(C88(I,14).EQ.'4M') X88(I,15)='-1 -1  1  1  1 -1  1'
IF(C88(I,14).EQ.'4N') X88(I,15)=' 1  1 -1  1  1 -1  1'
IF(C88(I,14).EQ.'4O') X88(I,15)='-1  1 -1  1  1 -1  1'
IF(C88(I,14).EQ.'4P') X88(I,15)=' 1 -1 -1  1  1 -1  1'
IF(C88(I,14).EQ.'4Q') X88(I,15)='-1 -1 -1  1  1 -1  1'
IF(C88(I,14).EQ.'4R') X88(I,15)=' 1  1  1 -1  1 -1  1'
IF(C88(I,14).EQ.'4S') X88(I,15)='-1  1  1 -1  1 -1  1'
IF(C88(I,14).EQ.'4T') X88(I,15)=' 1 -1  1 -1  1 -1  1'
IF(C88(I,14).EQ.'4U') X88(I,15)='-1 -1  1 -1  1 -1  1'
IF(C88(I,14).EQ.'4V') X88(I,15)=' 1  1 -1 -1  1 -1  1'
IF(C88(I,14).EQ.'4W') X88(I,15)='-1  1 -1 -1  1 -1  1'
IF(C88(I,14).EQ.'4Y') X88(I,15)=' 1 -1 -1 -1  1 -1  1'
IF(C88(I,14).EQ.'5A') X88(I,15)='-1 -1 -1 -1  1 -1  1'
IF(C88(I,14).EQ.'5B') X88(I,15)=' 1  1  1  1 -1 -1  1'
IF(C88(I,14).EQ.'5Y') X88(I,15)='-1  1  1  1 -1 -1  1'
IF(C88(I,14).EQ.'6A') X88(I,15)=' 1 -1  1  1 -1 -1  1'
IF(C88(I,14).EQ.'6B') X88(I,15)='-1 -1  1  1 -1 -1  1'
IF(C88(I,14).EQ.'6C') X88(I,15)=' 1  1 -1  1 -1 -1  1'
IF(C88(I,14).EQ.'6D') X88(I,15)='-1  1 -1  1 -1 -1  1'
IF(C88(I,14).EQ.'6E') X88(I,15)=' 1 -1 -1  1 -1 -1 -1'
IF(C88(I,14).EQ.'6F') X88(I,15)='-1 -1 -1  1 -1 -1 -1'
IF(C88(I,14).EQ.'6G') X88(I,15)=' 1  1  1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'6H') X88(I,15)='-1  1  1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'6I') X88(I,15)=' 1 -1  1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'6J') X88(I,15)='-1 -1  1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'6Y') X88(I,15)=' 1  1 -1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'7A') X88(I,15)='-1  1 -1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'7B') X88(I,15)=' 1 -1 -1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'7C') X88(I,15)='-1 -1 -1 -1 -1 -1 -1'
IF(C88(I,14).EQ.'7D') X88(I,15)=' 1  1  1  1  1  1 -1'
IF(C88(I,14).EQ.'7E') X88(I,15)='-1  1  1  1  1  1 -1'
IF(C88(I,14).EQ.'7F') X88(I,15)=' 1 -1  1  1  1  1 -1'
IF(C88(I,14).EQ.'7G') X88(I,15)='-1 -1  1  1  1  1 -1'
IF(C88(I,14).EQ.'7Y') X88(I,15)=' 1  1 -1  1  1  1 -1'
IF(C88(I,14).EQ.'8A') X88(I,15)='-1  1 -1  1  1  1 -1'
IF(C88(I,14).EQ.'8B') X88(I,15)=' 1 -1 -1  1  1  1 -1'
IF(C88(I,14).EQ.'8C') X88(I,15)='-1 -1 -1  1  1  1 -1'
IF(C88(I,14).EQ.'8D') X88(I,15)=' 1  1  1 -1  1  1 -1'
IF(C88(I,14).EQ.'8E') X88(I,15)='-1  1  1 -1  1  1 -1'
IF(C88(I,14).EQ.'8F') X88(I,15)=' 1 -1  1 -1  1  1 -1'
IF(C88(I,14).EQ.'8G') X88(I,15)='-1 -1  1 -1  1  1 -1'
IF(C88(I,14).EQ.'8H') X88(I,15)=' 1  1 -1 -1  1  1 -1'
IF(C88(I,14).EQ.'8Y') X88(I,15)='-1  1 -1 -1  1  1 -1'
```

```
        IF(C88(I,14).EQ.'9A') X88(I,15)=' 1 -1 -1 -1  1  1 -1'
        IF(C88(I,14).EQ.'9B') X88(I,15)='-1 -1 -1 -1  1  1 -1'
        IF(C88(I,14).EQ.'9C') X88(I,15)=' 1  1  1  1 -1  1 -1'
        IF(C88(I,14).EQ.'9D') X88(I,15)='-1  1  1  1 -1  1 -1'
        IF(C88(I,14).EQ.'9E') X88(I,15)=' 1 -1  1  1 -1  1 -1'
        IF(C88(I,14).EQ.'9F') X88(I,15)='-1 -1  1  1 -1  1 -1'
        IF(C88(I,14).EQ.'9G') X88(I,15)=' 1  1 -1  1 -1  1 -1'
        IF(C88(I,14).EQ.'9H') X88(I,15)='-1  1 -1  1 -1  1 -1'
        IF(C88(I,14).EQ.'9I') X88(I,15)=' 1 -1 -1  1 -1  1 -1'
        IF(C88(I,14).EQ.'9Y') X88(I,15)='-1 -1 -1  1 -1  1 -1'
        IF(C88(I,14).EQ.'YY') X88(I,15)=' 1  1  1 -1 -1  1 -1'
        IF(C88(I,14).EQ.'ZZ') X88(I,15)='-1  1  1 -1 -1  1 -1'


**********************************
***** DAFSC PREFIX **********
**********************************

        DAFSC = C88(I,15)
        FIRST = DAFSC(1:1)
        IF (FIRST.EQ.'A') THEN
          X88(I,16)=' 1  1  1'
        ELSE IF (FIRST.EQ.'S') THEN
          X88(I,16)='-1  1  1'
        ELSE IF (FIRST.EQ.'F') THEN
          X88(I,16)=' 1 -1  1'
        ELSE IF(FIRST.EQ.'K') THEN
          X88(I,16)='-1 -1  1'
        ELSE IF(FIRST.EQ.'M') THEN
          X88(I,16)=' 1  1 -1'
        ELSE IF(FIRST.EQ.'N') THEN
          X88(I,16)='-1  1 -1'
        ELSE IF(FIRST.EQ.'X') THEN
          X88(I,16)=' 1 -1 -1'
        ELSE
          X88(I,16)='-1 -1 -1'
          GO TO  150
        END IF

**********************************
********** DASFC DIGIT **********
**********************************
        GO TO  160
  150    DIGIT = C88(I,15)
        SECON = DIGIT(1:4)
        IF(SECON.EQ.'1025') THEN
          X88(I,17)=' 1  1  1  1  1'
        ELSE IF(SECON.EQ.'1035') THEN
          X88(I,17)='-1  1  1  1  1'
        ELSE IF(SECON.EQ.'1045') THEN
          X88(I,17)=' 1 -1  1  1  1'
```

```
      ELSE IF(SECON.EQ.'1055') THEN
        X88(I,17)='-1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1065') THEN
        X88(I,17)=' 1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1115') THEN
        X88(I,17)='-1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1145') THEN
        X88(I,17)=' 1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1165') THEN
        X88(I,17)='-1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1235') THEN
        X88(I,17)=' 1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1315') THEN
        X88(I,17)='-1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1325') THEN
        X88(I,17)=' 1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1335') THEN
        X88(I,17)='-1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1355') THEN
        X88(I,17)=' 1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1365') THEN
        X88(I,17)='-1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1406') THEN
        X88(I,17)=' 1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1415') THEN
        X88(I,17)='-1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1425') THEN
        X88(I,17)=' 1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1435') THEN
        X88(I,17)='-1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1445') THEN
        X88(I,17)=' 1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1455') THEN
        X88(I,17)='-1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1465') THEN
       X88(I,17)=' 1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1475') THEN
       X88(I,17)='-1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1485') THEN
       X88(I,17)=' 1 -1 -1  1 -1'
      ELSE IF(SECON.EQ.'1495') THEN
       X88(I,17)='-1 -1 -1  1 -1'
      ELSE
       X88(I,17)='-1 -1 -1 -1 -1'
      ENDIF
      GO TO  180

160    DIGIT = C88(I,15)
      SECON = DIGIT(2:4)
      IF(SECON.EQ.'1025') THEN
        X88(I,17)=' 1  1  1  1  1'
```

```
ELSE IF(SECON.EQ.'1035') THEN
  X88(I,17)='-1  1  1  1  1'
ELSE IF(SECON.EQ.'1045') THEN
  X88(I,17)=' 1 -1  1  1  1'
ELSE IF(SECON.EQ.'1055') THEN
  X88(I,17)='-1 -1  1  1  1'
ELSE IF(SECON.EQ.'1065') THEN
  X88(I,17)=' 1  1 -1  1  1'
ELSE IF(SECON.EQ.'1115') THEN
  X88(I,17)='-1  1 -1  1  1'
ELSE IF(SECON.EQ.'1145') THEN
  X88(I,17)=' 1 -1 -1  1  1'
ELSE IF(SECON.EQ.'1165') THEN
  X88(I,17)='-1 -1 -1  1  1'
ELSE IF(SECON.EQ.'1235') THEN
  X88(I,17)=' 1  1  1 -1  1'
ELSE IF(SECON.EQ.'1315') THEN
  X88(I,17)='-1  1  1 -1  1'
ELSE IF(SECON.EQ.'1325') THEN
  X88(I,17)=' 1 -1  1 -1  1'
ELSE IF(SECON.EQ.'1335') THEN
  X88(I,17)='-1 -1  1 -1  1'
ELSE IF(SECON.EQ.'1355') THEN
  X88(I,17)=' 1  1 -1 -1  1'
ELSE IF(SECON.EQ.'1365') THEN
  X88(I,17)='-1  1 -1 -1  1'
ELSE IF(SECON.EQ.'1406') THEN
  X88(I,17)=' 1 -1 -1 -1  1'
ELSE IF(SECON.EQ.'1415') THEN
  X88(I,17)='-1 -1 -1 -1  1'
ELSE IF(SECON.EQ.'1425') THEN
  X88(I,17)=' 1  1  1  1 -1'
ELSE IF(SECON.EQ.'1435') THEN
  X88(I,17)='-1  1  1  1 -1'
ELSE IF(SECON.EQ.'1445') THEN
  X88(I,17)=' 1 -1  1  1 -1'
ELSE IF(SECON.EQ.'1455') THEN
  X88(I,17)='-1 -1  1  1 -1'
ELSE IF(SECON.EQ.'1465') THEN
  X88(I,17)=' 1  1 -1  1 -1'
ELSE IF(SECON.EQ.'1475') THEN
  X88(I,17)='-1  1 -1  1 -1'
ELSE IF(SECON.EQ.'1485') THEN
  X88(I,17)=' 1 -1 -1  1 -1'
ELSE IF(SECON.EQ.'1495') THEN
  X88(I,17)='-1 -1 -1  1 -1'
ELSE
  X88(I,17)='-1 -1 -1 -1 -1'
ENDIF
```

```
*********************************
********** PAFSC PREFIX **********
*********************************

  180    PAFSC = C88(I,16)
        FIRST = PAFSC(1:1)
        IF (FIRST.EQ.'A') THEN
          X88(I,18)=' 1  1  1'
        ELSE IF (FIRST.EQ.'S') THEN
          X88(I,18)='-1  1  1'
        ELSE IF (FIRST.EQ.'F') THEN
          X88(I,18)=' 1 -1  1'
        ELSE IF(CFIRST.EQ.'K') THEN
          X88(I,18)='-1 -1  1'
        ELSE IF(FIRST.EQ.'M') THEN
          X88(I,18)=' 1  1 -1'
        ELSE IF(FIRST.EQ.'N') THEN
          X88(I,18)='-1  1 -1'
        ELSE IF(FIRST.EQ.'X') THEN
          X88(I,18)=' 1 -1 -1'
        ELSE
          X88(I,18)='-1 -1 -1'
          GO TO  120
        END IF

*********************************
********** PASFC DIGIT **********
*********************************
        GO TO  130
  120    DIGIT = C88(I,15)
        SECON = DIGIT(1:4)
        IF(SECON.EQ.'1025') THEN
          X88(I,19)=' 1  1  1  1  1'
        ELSE IF(SECON.EQ.'1035') THEN
          X88(I,19)='-1  1  1  1  1'
        ELSE IF(SECON.EQ.'1045') THEN
          X88(I,19)=' 1 -1  1  1  1'
        ELSE IF(SECON.EQ.'1055') THEN
          X88(I,19)='-1 -1  1  1  1'
        ELSE IF(SECON.EQ.'1065') THEN
          X88(I,19)=' 1  1 -1  1  1'
        ELSE IF(SECON.EQ.'1115') THEN
          X88(I,19)='-1  1 -1  1  1'
        ELSE IF(SECON.EQ.'1145') THEN
          X88(I,19)=' 1 -1 -1  1  1'
        ELSE IF(SECON.EQ.'1165') THEN
          X88(I,19)='-1 -1 -1  1  1'
        ELSE IF(SECON.EQ.'1235') THEN
          X88(I,19)=' 1  1  1 -1  1'
        ELSE IF(SECON.EQ.'1315') THEN
          X88(I,19)='-1  1  1 -1  1'
```

```
      ELSE IF(SECON.EQ.'1325') THEN
        X88(I,19)=' 1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1335') THEN
        X88(I,19)='-1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1355') THEN
        X88(I,19)=' 1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1365') THEN
        X88(I,19)='-1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1406') THEN
        X88(I,19)=' 1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1415') THEN
        X88(I,19)='-1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1425') THEN
        X88(I,19)=' 1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1435') THEN
        X88(I,19)='-1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1445') THEN
        X88(I,19)=' 1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1455') THEN
        X88(I,19)='-1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1465') THEN
        X88(I,19)=' 1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1475') THEN
        X88(I,19)='-1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1485') THEN
        X88(I,19)=' 1 -1 -1  1 -1'
      ELSE IF(SECON.EQ.'1495') THEN
        X88(I,19)='-1 -1 -1  1 -1'
      ELSE
        X88(I,19)='-1 -1 -1 -1 -1'
      END IF
      GO TO  140

130    DIGIT = C88(I,15)
      SECON = DIGIT(2:4)
      IF(SECON.EQ.'1025') THEN
        X88(I,19)=' 1  1  1  1  1'
      ELSE IF(SECON.EQ.'1035') THEN
        X88(I,19)='-1  1  1  1  1'
      ELSE IF(SECON.EQ.'1045') THEN
        X88(I,19)=' 1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1055') THEN
        X88(I,19)='-1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1065') THEN
        X88(I,19)=' 1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1115') THEN
        X88(I,19)='-1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1145') THEN
        X88(I,19)=' 1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1165') THEN
        X88(I,19)='-1 -1 -1  1  1'
```

```
        ELSE IF(SECON.EQ.'1235') THEN
          X88(I,19)=' 1  1  1 -1  1'
        ELSE IF(SECON.EQ.'1315') THEN
          X88(I,19)='-1  1  1 -1  1'
        ELSE IF(SECON.EQ.'1325') THEN
          X88(I,19)=' 1 -1  1 -1  1'
        ELSE IF(SECON.EQ.'1335') THEN
          X88(I,19)='-1 -1  1 -1  1'
        ELSE IF(SECON.EQ.'1355') THEN
          X88(I,19)=' 1  1 -1 -1  1'
        ELSE IF(SECON.EQ.'1365') THEN
          X88(I,19)='-1  1 -1 -1  1'
        ELSE IF(SECON.EQ.'1406') THEN
          X88(I,19)=' 1 -1 -1 -1  1'
        ELSE IF(SECON.EQ.'1415') THEN
          X88(I,19)='-1 -1 -1 -1  1'
        ELSE IF(SECON.EQ.'1425') THEN
          X88(I,19)=' 1  1  1  1 -1'
        ELSE IF(SECON.EQ.'1435') THEN
          X88(I,19)='-1  1  1  1 -1'
        ELSE IF(SECON.EQ.'1445') THEN
          X88(I,19)=' 1 -1  1  1 -1'
        ELSE IF(SECON.EQ.'1455') THEN
          X88(I,19)='-1 -1  1  1 -1'
        ELSE IF(SECON.EQ.'1465') THEN
          X88(I,19)=' 1  1 -1  1 -1'
        ELSE IF(SECON.EQ.'1475') THEN
          X88(I,19)='-1  1 -1  1 -1'
        ELSE IF(SECON.EQ.'1485') THEN
          X88(I,19)=' 1 -1 -1  1 -1'
        ELSE IF(SECON.EQ.'1495') THEN
          X88(I,19)='-1 -1 -1  1 -1'
        ELSE
          X88(I,19)='-1 -1 -1 -1 -1'
        END IF


*****************************
********** PRIORSV **********
*****************************

140     IF(C88(I,17).EQ.'0') X88(I,20)='-1  1'
        IF(C88(I,17).EQ.'1') X88(I,20)=' 1  1'
        IF(C88(I,17).EQ.'2') X88(I,20)='-1 -1'
        IF(C88(I,17).EQ.'3') X88(I,20)=' 1 -1'

*****************************
********** SOC **********
*****************************

        IF(C88(I,18).EQ.'A') X88(I,21)='-1  1  1'
```

```
        IF(C88(I,18).EQ.'B') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'C') X88(I,21)='-1  1 -1'
        IF(C88(I,18).EQ.'D') X88(I,21)=' 1  1  1'
        IF(C88(I,18).EQ.'E') X88(I,21)='-1 -1  1'
        IF(C88(I,18).EQ.'F') X88(I,21)=' 1  1 -1'
        IF(C88(I,18).EQ.'G') X88(I,21)=' 1 -1  1'
        IF(C88(I,18).EQ.'H') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'I') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'J') X88(I,21)='-1 -1 -1'
        IF(C88(I,18).EQ.'K') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'L') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'M') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'N') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'O') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'P') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'Q') X88(I,21)=' 1 -1 -1'
        IF(C88(I,18).EQ.'R') X88(I,21)='-1  1 -1'
        IF(C88(I,18).EQ.'S') X88(I,21)='-1 -1 -1'
        IF(C88(I,18).EQ.'T') X88(I,21)='-1  1 -1'
        IF(C88(I,18).EQ.'U') X88(I,21)='-1 -1 -1'
        IF(C88(I,18).EQ.'V') X88(I,21)='-1  1 -1'
        IF(C88(I,18).EQ.'W') X88(I,21)='-1 -1 -1'
        IF(C88(I,18).EQ.'X') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'Y') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'Z') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'1') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'2') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'3') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'4') X88(I,21)='-1  1  1'
        IF(C88(I,18).EQ.'5') X88(I,21)='-1  1  1'


**************************
********** RACE **********
**************************

        IF(C88(I,19).EQ.'C') X88(I,22)='-1 -1 -1'
        IF(C88(I,19).EQ.'M') X88(I,22)='-1  1  1'
        IF(C88(I,19).EQ.'N') X88(I,22)=' 1 -1  1'
        IF(C88(I,19).EQ.'R') X88(I,22)='-1 -1  1'
        IF(C88(I,19).EQ.'X') X88(I,22)=' 1  1 -1'
        IF(C88(I,19).EQ.'Z') X88(I,22)=' 1  1 -1'

**************************
********** COMP **********
**************************

        IF(C88(I,20).EQ.'R') X88(I,23)='-1 -1'
        IF(C88(I,20).EQ.'V') X88(I,23)=' 1 -1'
        IF((C88(I,20).NE.'R').AND.(C88(I,22).NE.'V')) X88(I,23)='-1  1'
```

```
*************************
********** SEX **********
*************************

       IF(C88(I,21).EQ.'M') X88(I,24)='-1'
       IF(C88(I,21).EQ.'F') X88(I,24)=' 1'

******************************
********** FLYMONTH **********
******************************

       X88(I,25)=C88(I,22)

****************************
********** RETAIN **********
****************************
********* STAY = 1 *********
******** LEAVE = 0 *********
****************************

       X88(I,26)=C88(I,23)

 30   CONTINUE



***** THE FOLLOWING CODE RANDOMLY SELECTS VECTORS FOR THE TRAINING SET
***** AND WRITES THE VECTORS TO A FILE.

      DO 500 J=1,NR1
        WRITE(*,*)'INIT CHOICE',J
        CHOICE(J)=0
 500  CONTINUE

      DO 530 II =  1,N1TRAIN
      WRITE(*,*)'SELECTING 88 TRAIN',II
  14   CONTINUE
      TEMP=RNUNF()
      JJ=NINT(TEMP*NR1)
      IF ((JJ.LE.NR1).AND.(JJ.GT.0)) THEN
        DO 510 K =1,NR1
          IF (JJ.EQ.CHOICE(K)) GO TO  14
 510    CONTINUE
        DO 520 KK=1,N1COL
          XTR88(II,KK)=X88(JJ,KK)
 520    CONTINUE
        CHOICE(II)=JJ
      ELSE
        GO TO  14
      END IF
 530  CONTINUE
```

```
      DO 540 I=1,N1TRAIN
        WRITE(*,*)'WRITING 88 TRAIN',I
        NOISE=RNUNF()
        WRITE(13,200)XTR88(I,2),XTR88(I,4),XTR88(I,7),XTR88(I,8),
     ;XTR88(I,12),XTR88(I,14),XTR88(I,19),
     ;NOISE,XTR88(I,N1COL)
        WRITE(23,210)XTR88(I,2),XTR88(I,4),XTR88(I,7),XTR88(I,8),
     ;XTR88(I,12),XTR88(I,14),XTR88(I,19),
     ;NOISE,XTR88(I,N1COL)
 540  CONTINUE


***** THE FOLLOWING CODE RANDOMLY SELECTS VECTORS FOR THE TEST SET
***** AND WRITES THE VECTORS TO A FILE.

      DO 550 II=1,N1TEST
      WRITE(*,*)'SELECTING 88 TEST',II
  15   CONTINUE
      TEMP=RNUNF()
      JJ=NINT(TEMP*NR1)
      IF ((JJ.LE.NR1).AND.(JJ.GT.0)) THEN
        DO 560 K=1,NR1
          IF (JJ.EQ.CHOICE(K)) GO TO  15
 560    CONTINUE
        DO 570 KK =  1,N1COL
          XTS88(II,KK)=X88(JJ,KK)
 570    CONTINUE
        CHOICE(II+N1TRAIN)=JJ
      ELSE
        GO TO  15
      END IF
 550  CONTINUE

      DO 580 I =1,N1TEST
        WRITE(*,*)'WRITING 88 TEST',I
        NOISE=RNUNF()
        WRITE(14,200)XTS88(I,2),XTS88(I,4),XTS88(I,7),XTS88(I,8),
     ;XTS88(I,12),XTS88(I,14),XTS88(I,19),
     ;NOISE,XTS88(I,N1COL)
        WRITE(24,210)XTS88(I,2),XTS88(I,4),XTS88(I,7),XTS88(I,8),
     ;XTS88(I,12),XTS88(I,14),XTS88(I,19),
     ;NOISE,XTS88(I,N1COL)
 580  CONTINUE

***** THE FOLLOWING CODE READS THE VECTORS NOT SELECTED FOR THE TRAINING
***** OR TEST SET INTO A VALIDATION FILE.

      CNT=0
      DO 590 I=1,NR1
        DO 600 K=1,NR1
```

```
        IF (I.EQ.CHOICE(K)) GO TO 590
600     CONTINUE
      CNT=CNT+1
      WRITE(*,*)'SELECTING 88 VALIDATION',CNT
      DO 610 KK=1,N1COL
        XVL88(CNT,KK)=X88(I,KK)
610   CONTINUE
590   CONTINUE

      DO 620 I=1,CNT
        WRITE(*,*)'WRITING 88 VALIDATION',I
        NOISE=RNUNF()
        WRITE(15,200)XVL88(I,2),XVL88(I,4),XVL88(I,7),XVL88(I,8),
     ;XVL88(I,12),XVL88(I,14),XVL88(I,19),
     ;NOISE,XVL88(I,N1COL)
        WRITE(25,210)XVL88(I,2),XVL88(I,4),XVL88(I,7),XVL88(I,8),
     ;XVL88(I,12),XVL88(I,14),XVL88(I,19),
     ;NOISE,XVL88(I,N1COL)
620   CONTINUE

*****MULTILAYER PERCEPTRON OUTPUT FILE FORMAT
200   FORMAT(7(A20),F8.6,5X,A20)

*****DISCRIMINANT ANALYSIS OUTPUT FILE FORMAT
210   FORMAT(3(A20)/3(A20)/A20,F8.6,5X,A20)

*******************************************************************************
***************************** FY 89 DATA ******************************
*******************************************************************************

      DO 70 I=1,NR2

      WRITE(*,*)'FY89 DATA     ITERATION = ',I

*********************************
********** TAFMSD (YOS) **********
*********************************

        X89(I,1)=C89(I,1)

************************************************
********** ADSCDA **********************************
************************************************

        X89(I,2)=C89(I,2)

*********************************
********** MARSTAT **********
*****************************

        IF (C89(I,3).EQ.'M') THEN
```

```
          X89(I,3)='-1 -1'
        ELSE IF (C89(I,3).EQ.'S') THEN
          X89(I,3)=' 1 -1'
        ELSE IF (C89(I,3).EQ.'D') THEN
          X89(I,3)='-1  1'
        ELSE
          X89(I,3)=' 1  1'
        ENDIF

**************************
********** DEPN **********
**************************

        X89(I,4)=C89(I,4)
        IF (X89(I,4).EQ.' .') X89(I,4)=' 0'

**************************
********** RDTM **********
**************************

        IF (C89(I,5).EQ.'A') THEN
          X89(I,5)=' 1  1  1'
        ELSE IF (C89(I,5).EQ.'D') THEN
          X89(I,5)='-1  1  1'
        ELSE IF (C89(I,5).EQ.'E') THEN
          X89(I,5)=' 1 -1  1'
        ELSE IF (C89(I,5).EQ.'F') THEN
          X89(I,5)='-1 -1  1'
        ELSE IF (C89(I,5).EQ.'G') THEN
          X89(I,5)=' 1  1 -1'
        ELSE IF (C89(I,5).EQ.'H') THEN
          X89(I,5)='-1  1 -1'
        ELSE IF (C89(I,5).EQ.'J') THEN
          X89(I,5)=' 1 -1 -1'
        ELSE IF (C89(I,5).EQ.'K') THEN
          X89(I,5)='-1 -1 -1'
        ELSE IF (C89(I,5).EQ.'L') THEN
          X89(I,5)='-1 -1 -1'
        ELSE
          X89(I,5)='-1 -1 -1'
        END IF

*************************
********** RPI **********
*************************

        IF (C89(I,6).EQ.'0') X89(I,6)=' 1  1  1'
        IF (C89(I,6).EQ.'1') X89(I,6)='-1  1  1'
        IF (C89(I,6).EQ.'2') X89(I,6)='-1 -1 -1'
        IF (C89(I,6).EQ.'3') X89(I,6)=' 1 -1  1'
        IF (C89(I,6).EQ.'4') X89(I,6)='-1 -1  1'
```

```
        IF (C89(I,6).EQ.'5') X89(I,6)='-1 -1 -1'
        IF (C89(I,6).EQ.'6') X89(I,6)=' 1  1 -1'
        IF (C89(I,6).EQ.'7') X89(I,6)='-1  1 -1'
        IF (C89(I,6).EQ.'8') X89(I,6)=' 1 -1 -1'
        IF (C89(I,6).EQ.'9') X89(I,6)='-1 -1 -1'

*****************************
********** GRADEA ***********
*****************************

        X89(I,7)=C89(I,7)

*****************************
********** RETPROG **********
*****************************

        IF (C89(I,8).EQ.'1') THEN
          X89(I,8)='-1 -1'
        ELSE IF (C89(I,8).EQ.'2') THEN
          X89(I,8)='-1  1'
        ELSE IF (C89(I,8).EQ.'3') THEN
          X89(I,8)=' 1 -1'
        ELSE
          X89(I,8)=' 1  1'
        ENDIF

*************************
********** PME **********
*************************

        IF (C89(I,9).EQ.'A') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'B') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'C') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'D') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'E') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'F') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'G') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'H') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'J') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'K') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'L') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'M') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'N') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'P') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'Q') X89(I,9)='-1 -1'
        IF (C89(I,9).EQ.'R') X89(I,9)='-1 -1'
        IF (C89(I,9).EQ.'S') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'T') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'U') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'V') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'W') X89(I,9)=' 1  1'
```

```
        IF (C89(I,9).EQ.'X') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'Z') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'1') X89(I,9)='-1 -1'
        IF (C89(I,9).EQ.'2') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'3') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'4') X89(I,9)=' 1 -1'
        IF (C89(I,9).EQ.'5') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'6') X89(I,9)='-1  1'
        IF (C89(I,9).EQ.'7') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'8') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.'9') X89(I,9)=' 1  1'
        IF (C89(I,9).EQ.' ') X89(I,9)=' 1  1'

**********************************
********** DESIRED PME **********
**********************************
********* COMPLETED =  1 *********
******* NOT COMPLETED = -1 *******
**********************************

        IF (((X89(I,9).EQ.'-1 -1'  ).OR.(X89(I,9).EQ.'-1  1' ).OR.
     ;  (X89(I,9).EQ.' 1 -1' )).AND.(X89(I,1).LT.8112)) X89(I,10)=' 1'

       IF (((X89(I,9).EQ.'-1  1' ).OR.
     ;  (X89(I,9).EQ.' 1 -1' )).AND.(X89(I,1).LT.7512)) X89(I,10)=' 1'

       IF (X89(I,9).EQ.' 1 -1' ) X89(I,10)=' 1'

       IF (((X89(I,9).NE.'-1 -1'  ).AND.(X89(I,9).NE.'-1  1' ).AND.
     ;  (X89(I,9).NE.' 1 -1' )).AND.(X89(I,1).LT.8112)) X89(I,10)='-1'

       IF (((X89(I,9).NE.'-1  1' ).AND.
     ;  (X89(I,9).NE.' 1 -1' )).AND.(X89(I,1).LT.7512)) X89(I,10)='-1'

       IF ((X89(I,9).NE.' 1 -1' ).AND.(X89(I,1).LT.7112)) X89(I,10)='-1'

****************************
********** MAJCOM **********
****************************

        IF (C89(I,10).EQ.'OA') X89(I,11)=' 1  1  1  1'
        IF (C89(I,10).EQ.'OB') X89(I,11)='-1 -1 -1  1'
        IF (C89(I,10).EQ.'OC') X89(I,11)='-1 -1 -1 -1'
        IF (C89(I,10).EQ.'OD') X89(I,11)='-1  1  1  1'
        IF (C89(I,10).EQ.'OE') X89(I,11)='-1 -1 -1 -1'
        IF (C89(I,10).EQ.'OF') X89(I,11)=' 1 -1  1  1'
        IF (C89(I,10).EQ.'OG') X89(I,11)='-1 -1 -1 -1'
        IF (C89(I,10).EQ.'OH') X89(I,11)='-1 -1  1  1'
        IF (C89(I,10).EQ.'OI') X89(I,11)='-1 -1 -1 -1'
        IF (C89(I,10).EQ.'OJ') X89(I,11)=' 1  1 -1  1'
        IF (C89(I,10).EQ.'OK') X89(I,11)='-1  1 -1  1'
```

```
IF (C89(I,10).EQ.'0L') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'0M') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'0N') X89(I,11)=' 1 -1 -1  1'
IF (C89(I,10).EQ.'0O') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'0P') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'0Q') X89(I,11)='-1  1 -1 -1'
IF (C89(I,10).EQ.'0R') X89(I,11)=' 1  1  1 -1'
IF (C89(I,10).EQ.'0S') X89(I,11)='-1  1  1 -1'
IF (C89(I,10).EQ.'0T') X89(I,11)=' 1 -1  1 -1'
IF (C89(I,10).EQ.'0U') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'0V') X89(I,11)=' 1 -1 -1 -1'
IF (C89(I,10).EQ.'0X') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'0Y') X89(I,11)='-1 -1  1 -1'
IF (C89(I,10).EQ.'0Z') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'01') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'02') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'03') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'05') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'06') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'07') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'08') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'09') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'1S') X89(I,11)=' 1  1 -1 -1'
IF (C89(I,10).EQ.'1W') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'1X') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'2A') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2C') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'2E') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2F') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2G') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2H') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2I') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'2J') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'2K') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2L') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2M') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2N') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2P') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2R') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'2W') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'3C') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'3F') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'3G') X89(I,11)='-1  1  1  1'
IF (C89(I,10).EQ.'3H') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'3I') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'3R') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'3S') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'3V') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'34') X89(I,11)='-1 -1 -1 -1'
IF (C89(I,10).EQ.'77') X89(I,11)='-1 -1 -1  1'
IF (C89(I,10).EQ.'8J') X89(I,11)='-1 -1 -1 -1'
```

```
        IF (C89(I,10).EQ.'88') X89(I,11)='-1 -1 -1  1'
        IF (C89(I,10).EQ.'YY') X89(I,11)='-1 -1 -1 -1'

*****************************
********** DOB (AGE) ********
*****************************

        X89(I,12)=C89(I,11)

*****************************
********** PASCBPO **********
*****************************

        IF(C89(I,12).EQ.'AF') X89(I,13)=' 1 -1  1  1'
        IF(C89(I,12).EQ.'AH') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'AK') X89(I,13)='-1  1  1 -1'
        IF(C89(I,12).EQ.'AM') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'AT') X89(I,13)='-1  1  1  1'
        IF(C89(I,12).EQ.'AU') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'AX') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'AY') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'A2') X89(I,13)='-1  1  1 -1'
        IF(C89(I,12).EQ.'A3') X89(I,13)='-1  1  1 -1'
        IF(C89(I,12).EQ.'A4') X89(I,13)=' 1 -1 -1  1'
        IF(C89(I,12).EQ.'A5') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'A6') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'A7') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'A8') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'A9') X89(I,13)=' 1 -1 -1  1'
        IF(C89(I,12).EQ.'BD') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'BF') X89(I,13)='-1  1  1 -1'
        IF(C89(I,12).EQ.'BB') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'BH') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'BL') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'BN') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'BP') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'BV') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'BX') X89(I,13)=' 1 -1  1 -1'
        IF(C89(I,12).EQ.'B2') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'B3') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'B4') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'B5') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'B6') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'B7') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'B8') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'B9') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'CC') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'CD') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'CF') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'CH') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'CJ') X89(I,13)=' 1  1  1  1'
```

```
IF(C89(I,12).EQ.'CK') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'CL') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'CO') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'CP') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'CQ') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'CR') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'C2') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'C3') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'C4') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'C5') X89(I,13)='-1  1 -1  1'
IF(C89(I,12).EQ.'C6') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'C7') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'C8') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'C9') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'DF') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'DM') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'DW') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'D2') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'D3') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'D4') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'D5') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'D6') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'D7') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'D8') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'D9') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'EB') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'EC') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'ED') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'EE') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'EH') X89(I,13)=' 1 -1 -1  1'
IF(C89(I,12).EQ.'EJ') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'EL') X89(I,13)=' 1 -1 -1  1'
IF(C89(I,12).EQ.'EM') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'EP') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'E2') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'E3') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'E4') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'E5') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'E7') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'E8') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'E9') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'FA') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'FB') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'FC') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'FE') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'FF') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'FG') X89(I,13)='-1  1 -1  1'
IF(C89(I,12).EQ.'FH') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'FK') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'FM') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'FN') X89(I,13)=' 1  1  1  1'
```

```
IF(C89(I,12).EQ.'FQ') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'FR') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'FS') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'FT') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'FU') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'FV') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'FW') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'FX') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'F2') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'F3') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'F4') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'F6') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'F7') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'F8') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'GB') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'GC') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'GF') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'GM') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'GW') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'G1') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'G2') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'G3') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'G4') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'G5') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'G6') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'G7') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'G8') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'G9') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'HB') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'HH') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'HL') X89(I,13)='-1  1 -1  1'
IF(C89(I,12).EQ.'HP') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'HS') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'HV') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'H2') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'H3') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'H4') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'H6') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'H7') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'H8') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'H9') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'IC') X89(I,13)=' 1  1 -1  1'
IF(C89(I,12).EQ.'IK') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'IN') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'J2') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'J3') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'J4') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'J6') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'J7') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'J8') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'J9') X89(I,13)='-1 -1  1 -1'
```

```
IF(C89(I,12).EQ.'KB') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'KF') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'KH') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'KJ') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'KU') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'KV') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'KY') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'K2') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'K3') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'K4') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'K6') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'K7') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'K8') X89(I,13)=' 1 -1  1  1'
IF(C89(I,12).EQ.'K9') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'LA') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'LC') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'LD') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'LE') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'LJ') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'LK') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'LL') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'LP') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'LS') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'LU') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'LW') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'LY') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'L2') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'L3') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'L4') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'L5') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'L6') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'L7') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'L8') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'L9') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'MA') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'MB') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'MD') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'ME') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'MG') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'MH') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'MK') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'ML') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'MN') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'MO') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'MP') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'MT') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'MU') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'MW') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'MY') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'M2') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'M3') X89(I,13)=' 1  1  1 -1'
```

```
IF(C89(I,12).EQ.'M4') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'M5') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'M6') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'M7') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'M8') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'M9') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'NJ') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'NV') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'N2') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'N3') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'OD') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'OP') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'PD') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'PE') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'PF') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'PJ') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'PS') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'PV') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'RF') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'RJ') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'RM') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'RP') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'RX') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'R1') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'R2') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'R3') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'R4') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'R5') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'R6') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'R7') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'R8') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'R9') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'SB') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'SF') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'SJ') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'SM') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'SP') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'SQ') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'ST') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'S1') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'S2') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'S3') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'S4') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'S5') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'TA') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'TB') X89(I,13)=' 1 -1 -1  1'
IF(C89(I,12).EQ.'TC') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'TD') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'TE') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'TF') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'TG') X89(I,13)=' 1  1 -1 -1'
```

```
IF(C89(I,12).EQ.'TH') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'TI') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'TJ') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'TK') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'TL') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'TM') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'TN') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'TO') X89(I,13)='-1  1 -1  1'
IF(C89(I,12).EQ.'TP') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'TQ') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'TR') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'TS') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'TT') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'TV') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'TW') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'TX') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'TY') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'TZ') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'T1') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'T2') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'T3') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'T4') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'T5') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'T6') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'T7') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'T8') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'T9') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'UB') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'UC') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'UD') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'UE') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'UF') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'UG') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'UH') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'UI') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'UJ') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'UK') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'UL') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'UM') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'UN') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'UO') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'UP') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'UQ') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'UR') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'US') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'UT') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'UU') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'UV') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'UW') X89(I,13)=' 1 -1  1  1'
IF(C89(I,12).EQ.'UX') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'UY') X89(I,13)='-1  1  1 -1'
```

```
IF(C89(I,12).EQ.'UZ') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'U2') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'U3') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'U4') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'U5') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'U6') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'U7') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'U8') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'U9') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'VH') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'VQ') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'WA') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'WC') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'WD') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'WE') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'WF') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'WG') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'WH') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'WI') X89(I,13)=' 1 -1  1  1'
IF(C89(I,12).EQ.'WJ') X89(I,13)='-1  1 -1 -1'
IF(C89(I,12).EQ.'WK') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'WL') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'WM') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'WT') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'WU') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'WV') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'WZ') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'W1') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'W2') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'W3') X89(I,13)='-1 -1 -1  1'
IF(C89(I,12).EQ.'W4') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'W5') X89(I,13)=' 1  1  1 -1'
IF(C89(I,12).EQ.'W6') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'W7') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'W8') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'W9') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'YM') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'YY') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'YZ') X89(I,13)='-1  1  1  1'
IF(C89(I,12).EQ.'ZA') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'ZB') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'ZC') X89(I,13)=' 1 -1  1 -1'
IF(C89(I,12).EQ.'ZE') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'ZG') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'ZK') X89(I,13)='-1  1  1 -1'
IF(C89(I,12).EQ.'ZL') X89(I,13)=' 1  1 -1 -1'
IF(C89(I,12).EQ.'ZN') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'ZS') X89(I,13)='-1 -1  1 -1'
IF(C89(I,12).EQ.'1A') X89(I,13)=' 1  1  1  1'
IF(C89(I,12).EQ.'1B') X89(I,13)=' 1 -1 -1  1'
IF(C89(I,12).EQ.'1C') X89(I,13)='-1  1  1 -1'
```

```
        IF(C89(I,12).EQ.'1D') X89(I,13)=' 1 -1  1 -1'
        IF(C89(I,12).EQ.'1F') X89(I,13)='-1  1  1 -1'
        IF(C89(I,12).EQ.'1H') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'1K') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'1M') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'1P') X89(I,13)=' 1  1 -1  1'
        IF(C89(I,12).EQ.'1T') X89(I,13)=' 1 -1  1 -1'
        IF(C89(I,12).EQ.'1U') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'1V') X89(I,13)=' 1  1 -1 -1'
        IF(C89(I,12).EQ.'2B') X89(I,13)=' 1 -1  1 -1'
        IF(C89(I,12).EQ.'2D') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'7A') X89(I,13)=' 1  1  1 -1'
        IF(C89(I,12).EQ.'7C') X89(I,13)=' 1  1  1  1'
        IF(C89(I,12).EQ.'7S') X89(I,13)='-1 -1 -1  1'
        IF(C89(I,12).EQ.'7V') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'7Z') X89(I,13)='-1 -1  1 -1'
        IF(C89(I,12).EQ.'8C') X89(I,13)=' 1 -1  1 -1'
        IF(C89(I,12).EQ.'85') X89(I,13)='-1  1  1 -1 '
        IF(C89(I,12).EQ.'9C') X89(I,13)='-1  1  1 -1'
        IF(C89(I,12).EQ.'96') X89(I,13)=' 1  1 -1 -1'

*******************************
********** ACAD LVL **********
*******************************

        IF(C89(I,13).EQ.'A') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'B') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'C') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'D') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'E') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'F') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'G') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'H') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'I') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'J') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'N') THEN
          X89(I,14)=' 1  1  1'
        ELSE IF(C89(I,13).EQ.'O') THEN
          X89(I,14)='-1  1  1'
        ELSE IF(C89(I,13).EQ.'P') THEN
          X89(I,14)=' 1 -1  1'
```

```
        ELSE IF(C89(I,13).EQ.'Q') THEN
          X89(I,14)='-1 -1  1'
        ELSE IF(C89(I,13).EQ.'R') THEN
          X89(I,14)=' 1  1 -1'
        ELSE IF(C89(I,13).EQ.'S') THEN
          X89(I,14)=' 1 -1 -1'
        ELSE IF(C89(I,13).EQ.'T') THEN
          X89(I,14)=' 1 -1 -1'
        ELSE IF(C89(I,13).EQ.'U') THEN
          X89(I,14)=' 1 -1 -1'
        ELSE IF(C89(I,13).EQ.'Y') THEN
          X89(I,14)='-1 -1 -1'
        ELSE
          X89(I,14)='-1 -1 -1'
        ENDIF

*********************************
********** ACAD SPEC **********
*********************************

        IF(C89(I,14).EQ.'0C') X89(I,15)=' 1  1  1  1  1  1  1'
        IF(C89(I,14).EQ.'0G') X89(I,15)='-1  1  1  1  1  1  1'
        IF(C89(I,14).EQ.'0I') X89(I,15)=' 1 -1  1  1  1  1  1'
        IF(C89(I,14).EQ.'0S') X89(I,15)='-1 -1  1  1  1  1  1'
        IF(C89(I,14).EQ.'0Y') X89(I,15)=' 1  1 -1  1  1  1  1'
        IF(C89(I,14).EQ.'1A') X89(I,15)='-1  1 -1  1  1  1  1'
        IF(C89(I,14).EQ.'1B') X89(I,15)=' 1 -1 -1  1  1  1  1'
        IF(C89(I,14).EQ.'1C') X89(I,15)='-1 -1 -1  1  1  1  1'
        IF(C89(I,14).EQ.'1Y') X89(I,15)=' 1  1  1 -1  1  1  1'
        IF(C89(I,14).EQ.'2A') X89(I,15)='-1  1  1 -1  1  1  1'
        IF(C89(I,14).EQ.'2B') X89(I,15)=' 1 -1  1 -1  1  1  1'
        IF(C89(I,14).EQ.'2C') X89(I,15)='-1 -1  1 -1  1  1  1'
        IF(C89(I,14).EQ.'2D') X89(I,15)=' 1  1 -1 -1  1  1  1'
        IF(C89(I,14).EQ.'2E') X89(I,15)='-1  1 -1 -1  1  1  1'
        IF(C89(I,14).EQ.'2F') X89(I,15)=' 1 -1 -1 -1  1  1  1'
        IF(C89(I,14).EQ.'2G') X89(I,15)='-1 -1 -1 -1  1  1  1'
        IF(C89(I,14).EQ.'2H') X89(I,15)=' 1  1  1  1 -1  1  1'
        IF(C89(I,14).EQ.'2I') X89(I,15)='-1  1  1  1 -1  1  1'
        IF(C89(I,14).EQ.'2K') X89(I,15)=' 1 -1  1  1 -1  1  1'
        IF(C89(I,14).EQ.'2Y') X89(I,15)='-1 -1  1  1 -1  1  1'
        IF(C89(I,14).EQ.'3A') X89(I,15)=' 1  1 -1  1 -1  1  1'
        IF(C89(I,14).EQ.'3B') X89(I,15)='-1  1 -1  1 -1  1  1'
        IF(C89(I,14).EQ.'3Y') X89(I,15)=' 1 -1 -1  1 -1  1  1'
        IF(C89(I,14).EQ.'4A') X89(I,15)='-1 -1 -1  1 -1  1  1'
        IF(C89(I,14).EQ.'4B') X89(I,15)=' 1  1  1 -1 -1  1  1'
        IF(C89(I,14).EQ.'4C') X89(I,15)='-1  1  1 -1 -1  1  1'
        IF(C89(I,14).EQ.'4D') X89(I,15)=' 1 -1  1 -1 -1  1  1'
        IF(C89(I,14).EQ.'4E') X89(I,15)='-1 -1  1 -1 -1  1  1'
        IF(C89(I,14).EQ.'4F') X89(I,15)=' 1  1 -1 -1 -1  1  1'
        IF(C89(I,14).EQ.'4G') X89(I,15)='-1  1 -1 -1 -1  1  1'
        IF(C89(I,14).EQ.'4H') X89(I,15)=' 1 -1 -1 -1 -1  1  1'
```

```
IF(C89(I,14).EQ.'4I') X89(I,15)='-1 -1 -1 -1 -1  1  1'
IF(C89(I,14).EQ.'4J') X89(I,15)=' 1  1  1  1  1 -1  1'
IF(C89(I,14).EQ.'4K') X89(I,15)='-1  1  1  1  1 -1  1'
IF(C89(I,14).EQ.'4L') X89(I,15)=' 1 -1  1  1  1 -1  1'
IF(C89(I,14).EQ.'4M') X89(I,15)='-1 -1  1  1  1 -1  1'
IF(C89(I,14).EQ.'4N') X89(I,15)=' 1  1 -1  1  1 -1  1'
IF(C89(I,14).EQ.'4O') X89(I,15)='-1  1 -1  1  1 -1  1'
IF(C89(I,14).EQ.'4P') X89(I,15)=' 1 -1 -1  1  1 -1  1'
IF(C89(I,14).EQ.'4Q') X89(I,15)='-1 -1 -1  1  1 -1  1'
IF(C89(I,14).EQ.'4R') X89(I,15)=' 1  1  1 -1  1 -1  1'
IF(C89(I,14).EQ.'4S') X89(I,15)='-1  1  1 -1  1 -1  1'
IF(C89(I,14).EQ.'4T') X89(I,15)=' 1 -1  1 -1  1 -1  1'
IF(C89(I,14).EQ.'4U') X89(I,15)='-1 -1  1 -1  1 -1  1'
IF(C89(I,14).EQ.'4V') X89(I,15)=' 1  1 -1 -1  1 -1  1'
IF(C89(I,14).EQ.'4W') X89(I,15)='-1  1 -1 -1  1 -1  1'
IF(C89(I,14).EQ.'4Y') X89(I,15)=' 1 -1 -1 -1  1 -1  1'
IF(C89(I,14).EQ.'5A') X89(I,15)='-1 -1 -1 -1  1 -1  1'
IF(C89(I,14).EQ.'5B') X89(I,15)=' 1  1  1  1 -1 -1  1'
IF(C89(I,14).EQ.'5Y') X89(I,15)='-1  1  1  1 -1 -1  1'
IF(C89(I,14).EQ.'6A') X89(I,15)=' 1 -1  1  1 -1 -1  1'
IF(C89(I,14).EQ.'6B') X89(I,15)='-1 -1  1  1 -1 -1  1'
IF(C89(I,14).EQ.'6C') X89(I,15)=' 1  1 -1  1 -1 -1  1'
IF(C89(I,14).EQ.'6D') X89(I,15)='-1  1 -1  1 -1 -1  1'
IF(C89(I,14).EQ.'6E') X89(I,15)=' 1 -1 -1  1 -1 -1 -1'
IF(C89(I,14).EQ.'6F') X89(I,15)='-1 -1 -1  1 -1 -1 -1'
IF(C89(I,14).EQ.'6G') X89(I,15)=' 1  1  1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'6H') X89(I,15)='-1  1  1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'6I') X89(I,15)=' 1 -1  1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'6J') X89(I,15)='-1 -1  1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'6Y') X89(I,15)=' 1  1 -1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'7A') X89(I,15)='-1  1 -1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'7B') X89(I,15)=' 1 -1 -1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'7C') X89(I,15)='-1 -1 -1 -1 -1 -1 -1'
IF(C89(I,14).EQ.'7D') X89(I,15)=' 1  1  1  1  1  1 -1'
IF(C89(I,14).EQ.'7E') X89(I,15)='-1  1  1  1  1  1 -1'
IF(C89(I,14).EQ.'7F') X89(I,15)=' 1 -1  1  1  1  1 -1'
IF(C89(I,14).EQ.'7G') X89(I,15)='-1 -1  1  1  1  1 -1'
IF(C89(I,14).EQ.'7Y') X89(I,15)=' 1  1 -1  1  1  1 -1'
IF(C89(I,14).EQ.'8A') X89(I,15)='-1  1 -1  1  1  1 -1'
IF(C89(I,14).EQ.'8B') X89(I,15)=' 1 -1 -1  1  1  1 -1'
IF(C89(I,14).EQ.'8C') X89(I,15)='-1 -1 -1  1  1  1 -1'
IF(C89(I,14).EQ.'8D') X89(I,15)=' 1  1  1 -1  1  1 -1'
IF(C89(I,14).EQ.'8E') X89(I,15)='-1  1  1 -1  1  1 -1'
IF(C89(I,14).EQ.'8F') X89(I,15)=' 1 -1  1 -1  1  1 -1'
IF(C89(I,14).EQ.'8G') X89(I,15)='-1 -1  1 -1  1  1 -1'
IF(C89(I,14).EQ.'8H') X89(I,15)=' 1  1 -1 -1  1  1 -1'
IF(C89(I,14).EQ.'8Y') X89(I,15)='-1  1 -1 -1  1  1 -1'
IF(C89(I,14).EQ.'9A') X89(I,15)=' 1 -1 -1 -1  1  1 -1'
IF(C89(I,14).EQ.'9B') X89(I,15)='-1 -1 -1 -1  1  1 -1'
IF(C89(I,14).EQ.'9C') X89(I,15)=' 1  1  1  1 -1  1 -1'
IF(C89(I,14).EQ.'9D') X89(I,15)='-1  1  1  1 -1  1 -1'
```

```
      IF(C89(I,14).EQ.'9E') X89(I,15)=' 1 -1  1  1 -1  1 -1'
      IF(C89(I,14).EQ.'9F') X89(I,15)='-1 -1  1  1 -1  1 -1'
      IF(C89(I,14).EQ.'9G') X89(I,15)=' 1  1 -1  1 -1  1 -1'
      IF(C89(I,14).EQ.'9H') X89(I,15)='-1  1 -1  1 -1  1 -1'
      IF(C89(I,14).EQ.'9I') X89(I,15)=' 1 -1 -1  1 -1  1 -1'
      IF(C89(I,14).EQ.'9Y') X89(I,15)='-1 -1 -1  1 -1  1 -1'
      IF(C89(I,14).EQ.'YY') X89(I,15)=' 1  1  1 -1 -1  1 -1'
      IF(C89(I,14).EQ.'ZZ') X89(I,15)='-1  1  1 -1 -1  1 -1'

**********************************
********** DAFSC PREFIX **********
**********************************

      DAFSC =C89(I,15)
      FIRST = DAFSC(1:1)
      IF (FIRST.EQ.'A') THEN
       X89(I,16)=' 1  1  1'
      ELSE IF (FIRST.EQ.'S') THEN
       X89(I,16)='-1  1  1'
      ELSE IF (FIRST.EQ.'F') THEN
       X89(I,16)=' 1 -1  1'
      ELSE IF(CFIRST.EQ.'K') THEN
       X89(I,16)='-1 -1  1'
      ELSE IF(FIRST.EQ.'M') THEN
       X89(I,16)=' 1  1 -1'
      ELSE IF(FIRST.EQ.'N') THEN
       X89(I,16)='-1  1 -1'
      ELSE IF(FIRST.EQ.'X') THEN
       X89(I,16)=' 1 -1 -1'
      ELSE
       X89(I,16)='-1 -1 -1'
       GO TO  151
      END IF

**********************************
********** DASFC DIGIT **********
**********************************
      GO TO  161
  151  DIGIT =C89(I,15)
      SECON = DIGIT(1:4)
      IF(SECON.EQ.'1025') THEN
       X89(I,17)=' 1  1  1  1  1'
      ELSE IF(SECON.EQ.'1035') THEN
       X89(I,17)='-1  1  1  1  1'
      ELSE IF(SECON.EQ.'1045') THEN
       X89(I,17)=' 1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1055') THEN
       X89(I,17)='-1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1065') THEN
       X89(I,17)=' 1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1115') THEN
```

```
       X89(I,17)='-1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1145') THEN
       X89(I,17)=' 1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1165') THEN
       X89(I,17)='-1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1235') THEN
       X89(I,17)=' 1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1315') THEN
       X89(I,17)='-1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1325') THEN
       X89(I,17)=' 1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1335') THEN
       X89(I,17)='-1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1355') THEN
       X89(I,17)=' 1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1365') THEN
       X89(I,17)='-1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1406') THEN
       X89(I,17)=' 1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1415') THEN
       X89(I,17)='-1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1425') THEN
       X89(I,17)=' 1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1435') THEN
       X89(I,17)='-1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1445') THEN
       X89(I,17)=' 1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1455') THEN
       X89(I,17)='-1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1465') THEN
      X89(I,17)=' 1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1475') THEN
      X89(I,17)='-1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1485') THEN
      X89(I,17)=' 1 -1 -1  1 -1'
      ELSE IF(SECON.EQ.'1495') THEN
      X89(I,17)='-1 -1 -1  1 -1'
      ELSE
      X89(I,17)='-1 -1 -1 -1 -1'
      ENDIF
      GO TO  181

161    DIGIT =C89(I,15)
      SECON = DIGIT(2:4)
      IF(SECON.EQ.'1025') THEN
       X89(I,17)=' 1  1  1  1  1'
      ELSE IF(SECON.EQ.'1035') THEN
       X89(I,17)='-1  1  1  1  1'
      ELSE IF(SECON.EQ.'1045') THEN
       X89(I,17)=' 1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1055') THEN
```

```
          X89(I,17)='-1 -1  1  1  1'
         ELSE IF(SECON.EQ.'1065') THEN
          X89(I,17)=' 1  1 -1  1  1'
         ELSE IF(SECON.EQ.'1115') THEN
          X89(I,17)='-1  1 -1  1  1'
         ELSE IF(SECON.EQ.'1145') THEN
          X89(I,17)=' 1 -1 -1  1  1'
         ELSE IF(SECON.EQ.'1165') THEN
          X89(I,17)='-1 -1 -1  1  1'
         ELSE IF(SECON.EQ.'1235') THEN
          X89(I,17)=' 1  1  1 -1  1'
         ELSE IF(SECON.EQ.'1315') THEN
          X89(I,17)='-1  1  1 -1  1'
         ELSE IF(SECON.EQ.'1325') THEN
          X89(I,17)=' 1 -1  1 -1  1'
         ELSE IF(SECON.EQ.'1335') THEN
          X89(I,17)='-1 -1  1 -1  1'
         ELSE IF(SECON.EQ.'1355') THEN
          X89(I,17)=' 1  1 -1 -1  1'
         ELSE IF(SECON.EQ.'1365') THEN
          X89(I,17)='-1  1 -1 -1  1'
         ELSE IF(SECON.EQ.'1406') THEN
          X89(I,17)=' 1 -1 -1 -1  1'
         ELSE IF(SECON.EQ.'1415') THEN
          X89(I,17)='-1 -1 -1 -1  1'
         ELSE IF(SECON.EQ.'1425') THEN
          X89(I,17)=' 1  1  1  1 -1'
         ELSE IF(SECON.EQ.'1435') THEN
          X89(I,17)='-1  1  1  1 -1'
         ELSE IF(SECON.EQ.'1445') THEN
          X89(I,17)=' 1 -1  1  1 -1'
         ELSE IF(SECON.EQ.'1455') THEN
          X89(I,17)='-1 -1  1  1 -1'
         ELSE IF(SECON.EQ.'1465') THEN
          X89(I,17)=' 1  1 -1  1 -1'
         ELSE IF(SECON.EQ.'1475') THEN
          X89(I,17)='-1  1 -1  1 -1'
         ELSE IF(SECON.EQ.'1485') THEN
          X89(I,17)=' 1 -1 -1  1 -1'
         ELSE IF(SECON.EQ.'1495') THEN
          X89(I,17)='-1 -1 -1  1 -1'
         ELSE
          X89(I,17)='-1 -1 -1 -1 -1'
         ENDIF


**********************************
********** PAFSC PREFIX **********
**********************************

  181    PAFSC =C89(I,16)
```

```
        X89(I,17)='-1 -1  1  1  1'
       ELSE IF(SECON.EQ.'1065') THEN
        X89(I,17)=' 1  1 -1  1  1'
       ELSE IF(SECON.EQ.'1115') THEN
        X89(I,17)='-1  1 -1  1  1'
       ELSE IF(SECON.EQ.'1145') THEN
        X89(I,17)=' 1 -1 -1  1  1'
       ELSE IF(SECON.EQ.'1165') THEN
        X89(I,17)='-1 -1 -1  1  1'
       ELSE IF(SECON.EQ.'1235') THEN
        X89(I,17)=' 1  1  1 -1  1'
       ELSE IF(SECON.EQ.'1315') THEN
        X89(I,17)='-1  1  1 -1  1'
       ELSE IF(SECON.EQ.'1325') THEN
        X89(I,17)=' 1 -1  1 -1  1'
       ELSE IF(SECON.EQ.'1335') THEN
        X89(I,17)='-1 -1  1 -1  1'
       ELSE IF(SECON.EQ.'1355') THEN
        X89(I,17)=' 1  1 -1 -1  1'
       ELSE IF(SECON.EQ.'1365') THEN
        X89(I,17)='-1  1 -1 -1  1'
       ELSE IF(SECON.EQ.'1406') THEN
        X89(I,17)=' 1 -1 -1 -1  1'
       ELSE IF(SECON.EQ.'1415') THEN
        X89(I,17)='-1 -1 -1 -1  1'
       ELSE IF(SECON.EQ.'1425') THEN
        X89(I,17)=' 1  1  1  1 -1'
       ELSE IF(SECON.EQ.'1435') THEN
        X89(I,17)='-1  1  1  1 -1'
       ELSE IF(SECON.EQ.'1445') THEN
        X89(I,17)=' 1 -1  1  1 -1'
       ELSE IF(SECON.EQ.'1455') THEN
        X89(I,17)='-1 -1  1  1 -1'
       ELSE IF(SECON.EQ.'1465') THEN
        X89(I,17)=' 1  1 -1  1 -1'
       ELSE IF(SECON.EQ.'1475') THEN
        X89(I,17)='-1  1 -1  1 -1'
       ELSE IF(SECON.EQ.'1485') THEN
        X89(I,17)=' 1 -1 -1  1 -1'
       ELSE IF(SECON.EQ.'1495') THEN
        X89(I,17)='-1 -1 -1  1 -1'
       ELSE
        X89(I,17)='-1 -1 -1 -1 -1'
       ENDIF


**********************************
********** PAFSC PREFIX **********
**********************************

  181    PAFSC =C89(I,16)
```

```
      FIRST = PAFSC(1:1)
      IF (FIRST.EQ.'A') THEN
       X89(I,18)=' 1  1  1'
      ELSE IF (FIRST.EQ.'S') THEN
       X89(I,18)='-1  1  1'
      ELSE IF (FIRST.EQ.'F') THEN
       X89(I,18)=' 1 -1  1'
      ELSE IF(CFIRST.EQ.'K') THEN
       X89(I,18)='-1 -1  1'
      ELSE IF(FIRST.EQ.'M') THEN
       X89(I,18)=' 1  1 -1'
      ELSE IF(FIRST.EQ.'N') THEN
       X89(I,18)='-1  1 -1'
      ELSE IF(FIRST.EQ.'X') THEN
       X89(I,18)=' 1 -1 -1'
      ELSE
       X89(I,18)='-1 -1 -1'
        GO TO  121
      END IF

**********************************
********** PASFC DIGIT **********
**********************************
      GO TO  131
  121   DIGIT =C89(I,15)
      SECON = DIGIT(1:4)
      IF(SECON.EQ.'1025') THEN
       X89(I,19)=' 1  1  1  1  1'
      ELSE IF(SECON.EQ.'1035') THEN
       X89(I,19)='-1  1  1  1  1'
      ELSE IF(SECON.EQ.'1045') THEN
       X89(I,19)=' 1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1055') THEN
       X89(I,19)='-1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1065') THEN
       X89(I,19)=' 1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1115') THEN
       X89(I,19)='-1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1145') THEN
       X89(I,19)=' 1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1165') THEN
       X89(I,19)='-1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1235') THEN
       X89(I,19)=' 1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1315') THEN
       X89(I,19)='-1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1325') THEN
       X89(I,19)=' 1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1335') THEN
       X89(I,19)='-1 -1  1 -1  1'
      ELSE IF(SECON.EQ.'1355') THEN
```

```
       X89(I,19)=' 1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1365') THEN
       X89(I,19)='-1  1 -1 -1  1'
      ELSE IF(SECON.EQ.'1406') THEN
       X89(I,19)=' 1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1415') THEN
       X89(I,19)='-1 -1 -1 -1  1'
      ELSE IF(SECON.EQ.'1425') THEN
       X89(I,19)=' 1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1435') THEN
       X89(I,19)='-1  1  1  1 -1'
      ELSE IF(SECON.EQ.'1445') THEN
       X89(I,19)=' 1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1455') THEN
       X89(I,19)='-1 -1  1  1 -1'
      ELSE IF(SECON.EQ.'1465') THEN
       X89(I,19)=' 1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1475') THEN
       X89(I,19)='-1  1 -1  1 -1'
      ELSE IF(SECON.EQ.'1485') THEN
       X89(I,19)=' 1 -1 -1  1 -1'
      ELSE IF(SECON.EQ.'1495') THEN
       X89(I,19)='-1 -1 -1  1 -1'
      ELSE
       X89(I,19)='-1 -1 -1 -1 -1'
      END IF
      GO TO  141

131    DIGIT =C89(I,15)
      SECON = DIGIT(2:4)
      IF(SECON.EQ.'1025') THEN
       X89(I,19)=' 1  1  1  1  1'
      ELSE IF(SECON.EQ.'1035') THEN
       X89(I,19)='-1  1  1  1  1'
      ELSE IF(SECON.EQ.'1045') THEN
       X89(I,19)=' 1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1055') THEN
       X89(I,19)='-1 -1  1  1  1'
      ELSE IF(SECON.EQ.'1065') THEN
       X89(I,19)=' 1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1115') THEN
       X89(I,19)='-1  1 -1  1  1'
      ELSE IF(SECON.EQ.'1145') THEN
       X89(I,19)=' 1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1165') THEN
       X89(I,19)='-1 -1 -1  1  1'
      ELSE IF(SECON.EQ.'1235') THEN
       X89(I,19)=' 1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1315') THEN
       X89(I,19)='-1  1  1 -1  1'
      ELSE IF(SECON.EQ.'1325') THEN
```

```
        X89(I,19)=' 1 -1  1 -1  1'
       ELSE IF(SECON.EQ.'1335') THEN
        X89(I,19)='-1 -1  1 -1  1'
       ELSE IF(SECON.EQ.'1355') THEN
        X89(I,19)=' 1  1 -1 -1  1'
       ELSE IF(SECON.EQ.'1365') THEN
        X89(I,19)='-1  1 -1 -1  1'
       ELSE IF(SECON.EQ.'1406') THEN
        X89(I,19)=' 1 -1 -1 -1  1'
       ELSE IF(SECON.EQ.'1415') THEN
        X89(I,19)='-1 -1 -1 -1  1'
       ELSE IF(SECON.EQ.'1425') THEN
        X89(I,19)=' 1  1  1  1 -1'
       ELSE IF(SECON.EQ.'1435') THEN
        X89(I,19)='-1  1  1  1 -1'
       ELSE IF(SECON.EQ.'1445') THEN
        X89(I,19)=' 1 -1  1  1 -1'
       ELSE IF(SECON.EQ.'1455') THEN
        X89(I,19)='-1 -1  1  1 -1'
       ELSE IF(SECON.EQ.'1465') THEN
        X89(I,19)=' 1  1 -1  1 -1'
       ELSE IF(SECON.EQ.'1475') THEN
        X89(I,19)='-1  1 -1  1 -1'
       ELSE IF(SECON.EQ.'1485') THEN
        X89(I,19)=' 1 -1 -1  1 -1'
       ELSE IF(SECON.EQ.'1495') THEN
        X89(I,19)='-1 -1 -1  1 -1'
       ELSE
        X89(I,19)='-1 -1 -1 -1 -1'
       END IF


*****************************
********** PRIORSV **********
*****************************

  141  IF(C89(I,17).EQ.'0') X89(I,20)='-1  1'
       IF(C89(I,17).EQ.'1') X89(I,20)=' 1  1'
       IF(C89(I,17).EQ.'2') X89(I,20)='-1 -1'
       IF(C89(I,17).EQ.'3') X89(I,20)=' 1 -1'

***************************
*********** SOC ***********
***************************

       IF(C89(I,18).EQ.'A') X89(I,21)='-1  1  1'
       IF(C89(I,18).EQ.'B') X89(I,21)='-1  1  1'
       IF(C89(I,18).EQ.'C') X89(I,21)='-1  1 -1'
       IF(C89(I,18).EQ.'D') X89(I,21)=' 1  1  1'
       IF(C89(I,18).EQ.'E') X89(I,21)='-1 -1  1'
       IF(C89(I,18).EQ.'F') X89(I,21)=' 1  1 -1'
```

```
      IF(C89(I,18).EQ.'G') X89(I,21)=' 1 -1  1'
      IF(C89(I,18).EQ.'H') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'I') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'J') X89(I,21)='-1 -1 -1'
      IF(C89(I,18).EQ.'K') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'L') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'M') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'N') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'O') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'P') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'Q') X89(I,21)=' 1 -1 -1'
      IF(C89(I,18).EQ.'R') X89(I,21)='-1  1 -1'
      IF(C89(I,18).EQ.'S') X89(I,21)='-1 -1 -1'
      IF(C89(I,18).EQ.'T') X89(I,21)='-1  1 -1'
      IF(C89(I,18).EQ.'U') X89(I,21)='-1 -1 -1'
      IF(C89(I,18).EQ.'V') X89(I,21)='-1  1 -1'
      IF(C89(I,18).EQ.'W') X89(I,21)='-1 -1 -1'
      IF(C89(I,18).EQ.'X') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'Y') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'Z') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'1') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'2') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'3') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'4') X89(I,21)='-1  1  1'
      IF(C89(I,18).EQ.'5') X89(I,21)='-1  1  1'


**************************
********** RACE **********
**************************

      IF(C89(I,19).EQ.'C') X89(I,22)='-1 -1 -1'
      IF(C89(I,19).EQ.'M') X89(I,22)='-1  1  1'
      IF(C89(I,19).EQ.'N') X89(I,22)=' 1 -1  1'
      IF(C89(I,19).EQ.'R') X89(I,22)='-1 -1  1'
      IF(C89(I,19).EQ.'X') X89(I,22)=' 1  1 -1'
      IF(C89(I,19).EQ.'Z') X89(I,22)=' 1  1 -1'

**************************
********** COMP **********
**************************

      IF(C89(I,20).EQ.'R') X89(I,23)='-1 -1'
      IF(C89(I,20).EQ.'V') X89(I,23)=' 1 -1'
      IF((C89(I,20).NE.'R').AND.(C89(I,20).NE.'V')) X89(I,23)='-1  1'

**************************
********** SEX **********
**************************

      IF(C89(I,21).EQ.'M') X89(I,24)='-1'
```

```
       IF(C89(I,21).EQ.'F') X89(I,24)=' 1'

*****************************
********** FLYMONTH **********
*****************************

       X89(I,25)=C89(I,22)

*****************************
********** RETAIN **********
*****************************
********* STAY = 1 *********
******** LEAVE = 0 *********
*****************************


       X89(I,26)=C89(I,23)

 70   CONTINUE


***** THE FOLLOWING CODE RANDOMLY SELECTS VECTORS FOR THE TRAINING SET
***** AND WRITES THE VECTORS TO A FILE.

      DO 700 J=1,NR2
        WRITE(*,*)'INIT CHOICE',J
        CHOICE2(J)=0
 700  CONTINUE

      DO 730 II =  1,N2TRAIN
      WRITE(*,*)'SELECTING 89 TRAIN',II
 34   CONTINUE
      TEMP=RNUNF()
      JJ=NINT(TEMP*NR2)
      IF ((JJ.LE.NR2).AND.(JJ.GT.0)) THEN
        DO 710 K =1,NR2
          IF (JJ.EQ.CHOICE2(K)) GO TO 34
 710    CONTINUE
        DO 720 KK=1,N1COL
          XTR89(II,KK)=X89(JJ,KK)
 720    CONTINUE
        CHOICE2(II)=JJ
      ELSE
        GO TO 34
      END IF
 730  CONTINUE

      DO 740 I=1,N2TRAIN
        WRITE(*,*)'WRITING 89 TRAIN',I
        NOISE=RNUNF()
        WRITE(16,200)XTR89(I,2),XTR89(I,4),XTR89(I,7),XTR89(I,8),
```

```
     ;XTR89(I,12),XTR89(I,14),XTR89(I,19),
     ;NOISE,XTR89(I,N1COL)
        WRITE(26,210)XTR89(I,2),XTR89(I,4),XTR89(I,7),XTR89(I,8),
     ;XTR89(I,12),XTR89(I,14),XTR89(I,19),
     ;NOISE,XTR89(I,N1COL)
 740  CONTINUE

***** THE FOLLOWING CODE RANDOMLY SELECTS VECTORS FOR THE TEST SET
***** AND WRITES THE VECTORS TO A FILE.

      DO 750 II=1,N2TEST
      WRITE(*,*)'SELECTING 89 TEST',II
 35   CONTINUE
      TEMP=RNUNF()
      JJ=NINT(TEMP*NR2)
      IF ((JJ.LE.NR2).AND.(JJ.GT.0)) THEN
        DO 760 K=1,NR2
          IF (JJ.EQ.CHOICE2(K)) GO TO 35
 760    CONTINUE
        DO 770 KK =  1,N1COL
          XTS89(II,KK)=X89(JJ,KK)
 770    CONTINUE
        CHOICE2(II+N2TRAIN+1)=JJ
      ELSE
        GO TO 35
      END IF
 750  CONTINUE

      DO 780 I = 1,N2TEST
        WRITE(*,*)'WRITING 89 TEST',I
        NOISE=RNUNF()
        WRITE(17,200)XTS89(I,2),XTS89(I,4),XTS89(I,7),XTS89(I,8),
     ;XTS89(I,12),XTS89(I,14),XTS89(I,19),
     ;NOISE,XTS89(I,N1COL)
        WRITE(27,210)XTS89(I,2),XTS89(I,4),XTS89(I,7),XTS89(I,8),
     ;XTS89(I,12),XTS89(I,14),XTS89(I,19),
     ;NOISE,XTS89(I,N1COL)
 780  CONTINUE


***** THE FOLLOWING CODE READS THE VECTORS NOT SELECTED FOR THE TRAINING
***** OR TEST SET INTO A VALIDATION FILE.

      CNT=0
      DO 790 I=1,NR2
        DO 800 K=1,NR2
          IF (I.EQ.CHOICE2(K)) GO TO 790
 800    CONTINUE
      CNT=CNT+1
      WRITE(*,*)'SELECTING 89 VALIDATION',CNT
      DO 810 KK=1,N1COL
```

```
        XVL89(CNT,KK)=X89(I,KK)
  810   CONTINUE
  790   CONTINUE

        DO 820 I=1,CNT
          WRITE(*,*)'WRITING 89 VALIDATION',I
          NOISE=RNUNF()
          WRITE(18,200)XVL89(I,2),XVL89(I,4),XVL89(I,7),XVL89(I,8),
       ;XVL89(I,12),XVL89(I,14),XVL89(I,19),
       ;NOISE,XVL89(I,N1COL)
          WRITE(28,210)XVL89(I,2),XVL89(I,4),XVL89(I,7),XVL89(I,8),
       ;XVL89(I,12),XVL89(I,14),XVL89(I,19),
       ;NOISE,XVL89(I,N1COL)
  820   CONTINUE

        END

***** END OF PROGRAM*****
```

Appendix D. *SAS Logistic Regression Sample Program*

```
options linesize=78;
data pilots;
  input var1 1-4 var2 21-24 var3 41-42 var4 44-45
        #2 var5 1-2 var6 21-22 var7 24-25 var8 27-28
        var9 41-42 var10 44-45 var11 47-48
        #3 var12 1-2 var13 21-22 var14 24-25
        var15 41-42 var16 44-45
        #4 var17 1-2 var18 21-22 var19 24-25 var20 27-28
        var21 30-31 var22 41-44
        #5 var23 1-2 var24 4-5 var25 7-8 var26 10-11
        var27 21-22 var28 24-25 var29 27-28
        var30 41-42 var31 44-45 var32 47-48 var33 50-51
        var34 53-54 var35 56-57 var36 59-60
        #6 var37 1-2 var38 4-5 var39 7-8
        var40 21-22 var41 24-25 var42 27-28
        var43 30-31 var44 33-34
        var45 41-42 var46 44-45 var47 47-48
        #7 var48 1-2 var49 4-5 var50 7-8
        var51 10-11 var52 13-14
        var53 21-22 var54 24-25
        var55 41-42 var56 44-45 var57 47-48
        #8 var58 1-2 var59 4-5 var60 7-9
        var61 21-22 var62 24-25 var63 41-42
        #9 var64 1-4 var65 21-28 retain 33-34;

%include train88;

proc logistic data=pilots outest=param covout;
  model retain=var1 var2 var3 var4 var5 var6 var7 var8 var9 var10
               var11 var12 var13 var14 var15 var16 var17 var18
               var19 var20 var21 var22 var23 var24 var25 var26
               var27 var28 var29 var30 var31 var32 var33 var34
               var35 var36 var37 var38 var39 var40 var41 var42
               var43 var44 var45 var46 var47 var48 var49 var50
               var51 var52 var53 var54 var55 var56 var57 var58
               var59 var60 var61 var62 var63 var64 var65 / covb
               selection=stepwise
```

```
                slentry=0.3
                slstay=0.3
                ctable
                corrb;
  output out=pred p=phat lower=lcl upper=ucl;
  title 'Pilot Retention Analysis--All Factors';
run;
```

Appendix E. *SAS K-Nearest-Neighbor Sample Program*

```
options linesize=78;
data pilots;
  input var1 1-4 var2 21-24 var3 41-42 var4 44-45
        #2 var5 1-2 var6 21-22 var7 24-25 var8 27-28
        var9 41-42 var10 44-45 var11 47-48
        #3 var12 1-2 var13 21-22 var14 24-25
        var15 41-42 var16 44-45
        #4 var17 1-2 var18 21-22 var19 24-25 var20 27-28
        var21 30-31 var22 41-44
        #5 var23 1-2 var24 4-5 var25 7-8 var26 10-11
        var27 21-22 var28 24-25 var29 27-28
        var30 41-42 var31 44-45 var32 47-48 var33 50-51
        var34 53-54 var35 56-57 var36 59-60
        #6 var37 1-2 var38 4-5 var39 7-8
        var40 21-22 var41 24-25 var42 27-28
        var43 30-31 var44 33-34
        var45 41-42 var46 44-45 var47 47-48
        #7 var48 1-2 var49 4-5 var50 7-8
        var51 10-11 var52 13-14
        var53 21-22 var54 24-25
        var55 41-42 var56 44-45 var57 47-48
        #8 var58 1-2 var59 4-5 var60 7-9
        var61 21-22 var62 24-25 var63 41-42
        #9 var64 1-4 var65 21-28 retain 33-34;

%include train88;

data valid;
  input var1 1-4 var2 21-24 var3 41-42 var4 44-45
        #2 var5 1-2 var6 21-22 var7 24-25 var8 27-28
        var9 41-42 var10 44-45 var11 47-48
        #3 var12 1-2 var13 21-22 var14 24-25
        var15 41-42 var16 44-45
        #4 var17 1-2 var18 21-22 var19 24-25 var20 27-28
        var21 30-31 var22 41-44
        #5 var23 1-2 var24 4-5 var25 7-8 var26 10-11
        var27 21-22 var28 24-25 var29 27-28
```

```
        var30 41-42 var31 44-45 var32 47-48 var33 50-51
        var34 53-54 var35 56-57 var36 59-60
        #6 var37 1-2 var38 4-5 var39 7-8
        var40 21-22 var41 24-25 var42 27-28
        var43 30-31 var44 33-34
        var45 41-42 var46 44-45 var47 47-48
        #7 var48 1-2 var49 4-5 var50 7-8
        var51 10-11 var52 13-14
        var53 21-22 var54 24-25
        var55 41-42 var56 44-45 var57 47-48
        #8 var58 1-2 var59 4-5 var60 7-9
        var61 21-22 var62 24-25 var63 41-42
        #9 var64 1-4 var65 21-28 retain 33-34;

%include val88;

proc discrim data=pilots crossvalidate testlist method=npar k=7
     testdata=valid;
  class retain;
  priors proportional;
  testclass retain;
  var var1 var2 var3 var4 var5 var6 var7 var8 var9 var10
               var11 var12 var13 var14 var15 var16 var17 var18
               var19 var20 var21 var22 var23 var24 var25 var26
               var27 var28 var29 var30 var31 var32 var33 var34
               var35 var36 var37 var38 var39 var40 var41 var42
               var43 var44 var45 var46 var47 var48 var49 var50
               var51 var52 var53 var54 var55 var56 var57 var58
               var59 var60 var61 var62 var63 var64 var65;
  title 'Pilot Retention Analysis--All Factors';
run;
```

Appendix F. *SAS Stepwise Discriminant Analysis Sample Program*

```
options linesize=78;
data pilots;
  input var1 1-4 var2 21-24 var3 41-42 var4 44-45
        #2 var5 1-2 var6 21-22 var7 24-25 var8 27-28
        var9 41-42 var10 44-45 var11 47-48
        #3 var12 1-2 var13 21-22 var14 24-25
        var15 41-42 var16 44-45
        #4 var17 1-2 var18 21-22 var19 24-25 var20 27-28
        var21 30-31 var22 41-44
        #5 var23 1-2 var24 4-5 var25 7-8 var26 10-11
        var27 21-22 var28 24-25 var29 27-28
        var30 41-42 var31 44-45 var32 47-48 var33 50-51
        var34 53-54 var35 56-57 var36 59-60
        #6 var37 1-2 var38 4-5 var39 7-8
        var40 21-22 var41 24-25 var42 27-28
        var43 30-31 var44 33-34
        var45 41-42 var46 44-45 var47 47-48
        #7 var48 1-2 var49 4-5 var50 7-8
        var51 10-11 var52 13-14
        var53 21-22 var54 24-25
        var55 41-42 var56 44-45 var57 47-48
        #8 var58 1-2 var59 4-5 var60 7-9
        var61 21-22 var62 24-25 var63 41-42
        #9 var64 1-4 var65 21-28 retain 33-34;

%include train88;

proc stepdisc data=pilots short;
  class retain;
  var var1 var2 var3 var4 var5 var6 var7 var8 var9 var10
               var11 var12 var13 var14 var15 var16 var17 var18
               var19 var20 var21 var22 var23 var24 var25 var26
               var27 var28 var29 var30 var31 var32 var33 var34
               var35 var36 var37 var38 var39 var40 var41 var42
               var43 var44 var45 var46 var47 var48 var49 var50
               var51 var52 var53 var54 var55 var56 var57 var58
               var59 var60 var61 var62 var63 var64 var65;
```

```
  title 'Pilot Retention Analysis--All Factors';
run;
```

Appendix G. *Resulting Multilayer Perceptron Weights*

LOWER WEIGHTS: BETWEEN INPUT NODE 1 AND HIDDEN LAYER

3.744487	-5.661802	1.236460	1.922172	-2.029330	10.354836
6.098336	2.147858	4.904820	-11.311367	-1.917496	1.841209
9.352450	-3.212702	3.711942	2.021291	-12.909331	-1.363645
-5.616950	1.299951				

LOWER WEIGHTS: BETWEEN INPUT NODE 2 AND HIDDEN LAYER

1.347507	-0.970914	1.164620	1.239261	-6.904645	-0.094074
0.736928	0.922072	-0.114203	8.686078	-2.432656	-0.232804
-13.721185	-6.776404	-16.773806	0.401682	5.524731	-1.627905
5.696426	0.477476				

LOWER WEIGHTS: BETWEEN INPUT NODE 3 AND HIDDEN LAYER

1.919518	0.870390	1.480452	1.972719	-2.842570	0.711223
-1.605092	2.106697	-0.150365	-9.008704	-3.020168	1.646644
0.529753	-4.272395	5.095506	1.684121	-12.894631	-1.940293
-15.640748	1.351816				

LOWER WEIGHTS: BETWEEN INPUT BIAS AND HIDDEN LAYER

0.350284	3.616477	2.234319	1.472493	2.479678	-2.922721
-2.494206	1.428351	-1.127116	8.068166	1.193551	1.970396
4.1[illegible]7944	3.937369	8.015651	1.656948	18.750011	0.094867
5.196889	2.342147				

UPPER WEIGHTS: BETWEEN HIDDEN LAYER AND OUTPUT LAYER

OUTPUT NODE 1	OUTPUT NODE 2
1.622562	-2.391974
1.518672	-1.350263
0.701384	0.101410
1.359723	-0.511516

-3.708065	3.423710
4.255523	-4.189113
-1.464045	1.800414
0.722850	-1.476495
-0.951802	0.532561
-7.536643	7.476709
-1.830029	1.852042
-0.883811	0.523115
-5.536485	5.647514
-3.824516	4.102173
-6.893157	6.807547
-0.724301	0.604316
8.788891	-8.774087
-1.048357	0.822975
-11.893315	11.845622
0.077864	0.398423
0.755320	-0.879614

Bibliography

1. AF/XOOTW. "Rated Management Decision Support System." Report for Air Force Chief of Staff, 18 December 1990.
2. Bauer, Kenneth W. Class notes OPER 685, Applied Multivariate Data Analysis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, February 1992.
3. Defense Advanced Research Projects Agency (DARPA). *Neural Network Study* AFCEA International Press, Fairfax VI, November 1988.
4. Devijver, Pierre A. and Josef Kittler. *Pattern Recognition: A Statistical Approach.* Englewood Cliffs NJ: Prentice-Hall International, Inc., 1982.
5. Dillon, William R. and Matthew Goldstein. *Multivariate Analysis Methods and Applications.* New York:John Wiley and Sons, 1984.
6. Efron, Bradley. "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation," *Journal of the American Statistical Association, 78:* 316-320 (June 1983).
7. Falcon Research and Development (FRD). *Aircraft Fuel Tank Environment/Threat Model for Fire and Explosion Vulnerability Assessment (U), Volume II.* Aeronautical Systems Division Technical Report ASD-TR-77-19, Wright-Patterson AFB OH, March 1977.
8. Foley, Donald H. "Considerations of Sample and Feature Size," *IEEE Transactions on Information Theory, 18:* 618-626 (September 1972).
9. Giles, Lee C. and Tom Maxwell. "Learning invariance, and Generalization in high-order neural networks," *Applied Optics, 26:* 4972-4978 (1 December 1987).
10. Gotz, Glenn A. and John J. McCall. *A dynamic Retention Model for Air Force Officers: Theory and Estimates.* Contract F49620-82-C-0018. RAND Corporation, December 1984 (AD- A149736).
11. Gnanadesikan, Ramanathan. *et al.* "Discriminant Analysis and Clustering," *Statistical Science, 4:* 34-69 (December 1989).
12. Grant, Michael, Capt, USAF. Personal Correspondence. Directorate of Personnel Plans, Headquarters Air Force (AF/DPXA), Washington DC, 5 November 1990.
13. Guzowski, Major Bruce A. *A Methodology For Long-Term Forecasts of Air Force Pilot Retention Rates: A Management Perspective.* MS thesis, AFIT/GSM/LSR/90S-11. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1990 (AD-A229541).

14. Highleyman, W.H. "The Design and Analysis of Pattern Recognition Experiments," *The Bell System Technical Journal, 41:* 723-744 (March 1962).

15. Knight, Cpt Earl E. *Predicting Armor Piercing Incendiary Projectile Effects After Impacting Composite Material.* MS thesis, AFIT/GOR/ENS/92M-18. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1992.

16. Lippmann, Richard P. "Pattern Classification Using Neural Networks," *IEEE Communications Magazine, 27:* 47-62 (November 1989).

17. Mayerhofer, R.D. *Functioning Characteristics of Several Types of API Projectiles (U).* Ballistic Research Laboratories Report No. 216, Aberdeen Proving Ground MD, April 1974.

18. Minsky, Marvin Lee and Seymour Papert. *Perceptrons* (Expanded Edition). Cambridge: The MIT Press, 1988.

19. Pettit, Patricia A. *Incendiary Functioning Characteristics of Soviet API Projectiles Impacting Graphite/Epoxy Composite Panels, Final Report* WRDC-TR-90-3030, WRADC, Wright-Patterson AFB OH, April 1991.

20. Reynolds, Major Jon K. *A Response Surface Model for the Incendiary Functioning Characteristics of Soviet API Projectiles Impacting Graphite Epoxy Composite Panels.* MS thesis, AFIT/GOR/91M-13. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1990.

21. Rogers, Maj Steven K. and others. *An Introduction to Biological and Artificial Neural Networks.* Class handout distributed in EENG 817, Advanced Topics in Pattern Recognition. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, July 1991.

22. Ruck, Capt Dennis W. *Characterization of Multilayer Perceptrons and their Application to Multisensor Automatic Target Detection.* PhD dissertation. School of Engineering, Air Force Institute of Technology (AU), Wright- Patterson AFB OH, December 1990 (AD-A229035).

23. Ruck, Capt Dennis W. "Feature Selection Using a Multilayer Perceptron," *Journal of Neural Network Computing, 20:* 40-48 (Fall 1990).

24. SAS Institute Inc. *SAS User's Guide: Statistics, Version 5 Edition.* Cary, NC:SAS Institute Inc., 1985.

25. SAS Institute Inc. *SAS User's Guide: Statistics, Version 6 Edition.* Cary, NC:SAS Institute Inc., 1990.

26. Shigley, Elenor. *An Analysis of Factors Affecting the Career Plans of Military Nurses.* MS thesis, Naval Postgraduate School, Monterey CA, December 1988 (AD-A205928).

27. Simpson, Capt James R. *A Methodology for Forecasting Voluntary Retention Rates for Air Force Pilots.* MS thesis, AFIT/GOR/ENS/87D-18. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

28. Tarr, Capt Gregory, Draft PhD dissertation, School of Engineering, Air Force Institute of Technology (AU), Wright- Patterson AFB OH, November 1991.

29. Weiss, Sholom M. and Casimir A. Kulikowski. *Quarterly Report: Empirical Analysis and Refinement of Expert System Knowledge Bases.* Contract N00014-87-K-0398. New Brunswick NJ: Rutgers University Center for Expert Systems Research, 28 February 1989 (AD-A206226).

30. Whalen, Lieutenant Commander William P. *An Analysis of Factors Affecting the Retention of Medical Officers in the United States Navy.* MS thesis, Naval Postgraduate School, Monterey CA, December 1986 (AD-A178588).

31. White, Halbert. "Learning in Artificial Neural Networks: A Statistical Perspective," *Neural Computation, 1:* 425-464 (Fall 1989).

32. Wiggins, V.L. and others. *Applying Neural Networks to Air Force Personnel Analysis.* Contract F41689-88-D- 0251. Human Resources Directorate, Manpower and Personnel Research Division, Brooks Air Force Base, TX, April 1991.

Vita

Captain Lisa M. Belue was born on 21 May 1963 in Port Huron, Michigan. In 1981, she graduated from Port Huron High School and attended Michigan Technological University in Houghton, Michigan. In 1985, she graduated from Michigan Tech with a Bachelor of Science Degree in Mathematics. Her first assignment was as a personnel analyst for The Military Personnel Center at Randolph AFB, Texas. A subsequent assignment in San Antonio was as a future requirements analyst for the Directorate of Development Plans, Headquarters Human Systems Division, Brooks AFB. Captain Belue's next assignment was the Chief, Data Management for the Pacific Air Force's Weapon System Evaluation Program, Clark AB Republic of the Philippines supporting live fire air-to-air missile tests. Captain Belue entered the Air Force Institute of Technology in August of 1990.

Permanent address: 1455 Wadhams Road
Port Huron, Michigan
48060